techcamp.org.uk

B - Save our Robot!

What if our robot gets in trouble half way across the planet? Let's make a program so it can transmit S.O.S in Morse Code to let us know if its in trouble!

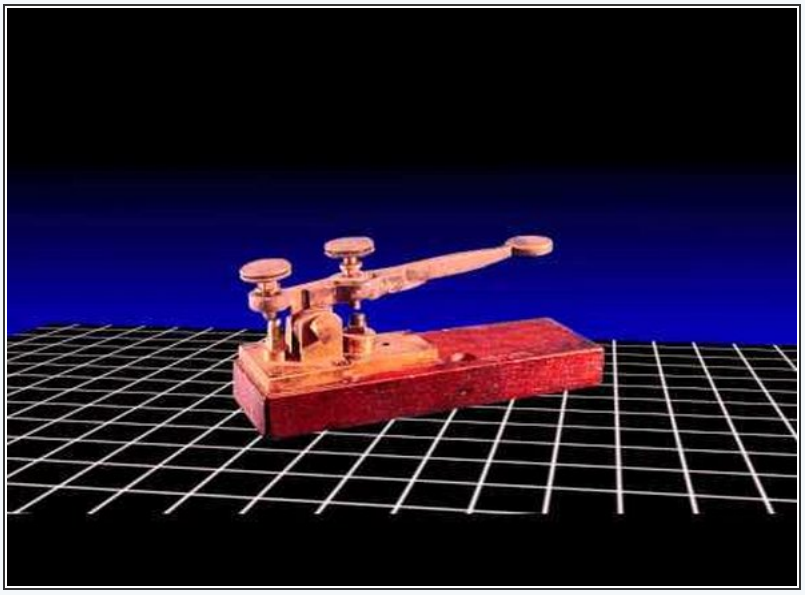
INTRODUCTION

What if our robot gets in trouble half way across the planet? Let's make a program so it can transmit S.O.S in Morse Code to let us know if its in trouble!

Step 1

Morse Code?

- Having red/green LEDs is a great start to communicating with our robot
- What if we want to communicate more than just red or green?
- **Morse Code** allows us to send any letter or number we like, just using a single light or buzzer!
- Watch the video to find out more.



Step 2

Letter S

- Let's write a program that will send the **letter S.**
- In the picture is a program to send **1 dot** with the red LED. For your **sleep()** lines, we suggest:
  - **100 milliseconds** for a dot
  - **1000 milliseconds** for a dash
- **Extend** the program in the picture to send **3 dots,** which is an **S** - your red LED module should still be plugged in to **P1** like the last lesson!

```
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  pin1.write_digital(1)
17  sleep(100)
18  pin1.write_digital(0)
19  sleep(100)
```

Step 3

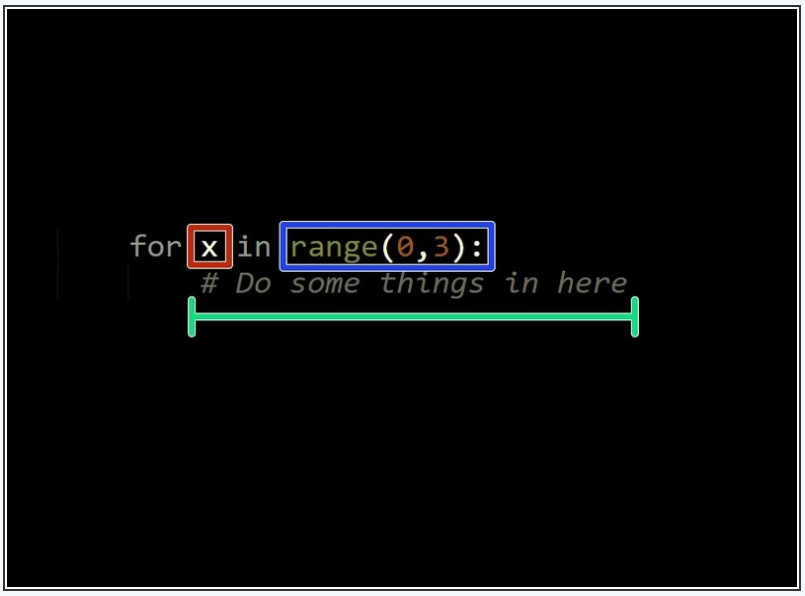
Make it Shorter

- Good programs always try **not to repeat code that does the same thing,** so the program is as small and efficient as possible.
- We could put the code for one dot in a **while True:** loop, but this would just do dots **forever** - not what we want!
- We are going to use a **for loop** to repeat something a set number of times.

Step 4

For Loops

- A **for loop** needs 3 things:
  - An **iterable** - this is something that Python uses to count how many times the loop has run. It can be called anything you like - we have called it **x** in this example.
  - A **range** to iterate over - a range in Python just creates a list of numbers, starting from the first one (0 in this case) and up to, but not including, the last one (3). So our range is **0,1,2.**
  - **Some code to run each time** - this is just like the while loop from before and can be as long or as complicated as you like!
- So this for loop starts with **x=0.** After the first loop run, x is increased to **1**. After the second it is increased to **2**, the code then runs a final time and the loop will finish. So this loop will run **3 times**.



⚠ Don't forget - just like a while loop, we need the **colon (:)** after the iterable and range, and any code you want to run in the loop needs to be **indented**.

Step 5

Use the for loop

- Put your code to send a **dot** with the red LED **inside** the for loop, so your robot sends an 's' with much less code.
- **Try changing the range,** and see how it affects how many dots get sent!

```
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 for x in range(0,3):
17     pin1.write_digital(1)
18     sleep(100)
19     pin1.write_digital(0)
20     sleep(100)
```

Step 6

Indentation

- Let's talk a bit more about **indentation** - we have already been doing it inside the for and while loops.
- **Indentation** is very important in programming - it makes it easy for **others** to read your code, and for **you** to read it later!
- You can **indent** a line by pressing the **TAB** key.
- In most languages, indentation is optional and only serves to make the code easier to read - however, in Python it is **not optional!**
- **Python uses indentation to define which lines of code are inside other things** - for example which lines of code should be repeated in a loop. See how all of the lines inside the for loop are indented the same?

⚠ You should always make sure Python code is indented correctly - **things won't work properly otherwise.**

```
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 for x in range(0,3):
17 →   pin1.write_digital(1)
18 →   sleep(100)
19 →   pin1.write_digital(0)
20 →   sleep(100)
```

Step 7

Gaps Between Letters

- You might have noticed when you were listening to the code, that **between the letters** we need **longer gaps** so you can tell when they **start and finish.**
- A time of **2 seconds** works well.
- Put your for loop that flashes an S inside a **while True:** loop, so it flashes S forever, and **add a wait block** so there is a **gap of seconds between each S.**

⚠ Make sure it is **indented properly** - the first line of the for loop should be indented **once** as it is inside the while loop, and the code inside the for loop should be indented **twice** - it is inside the while loop **and** the for loop!

```
13 # Invent! Code End
14 # Start your code below here!
15
16 while True:
17     for x in range(0,3):
18         pin1.write_digital(1)
19         sleep(100)
20         pin1.write_digital(0)
21         sleep(100)
```

Step 8

SOS Flasher

- Let's make a program to get our robot to flash **SOS** using an LED, in case it has a problem.
- Your program should flash the sequence of dots and dashes required for the letters SOS - for bonus points put it in a while loop to make it flash SOS **forever!**
- Try to shorten your program using **for loops** - you will need **3 separate loops**, or 2 if you're really clever!

Step 9

Buzzer SOS

Extension Challenge!

- Using lights for Morse Code is great for long distances, such as between two ships, but what if you are looking the other way when your robot is in trouble?
- Replace your LED module with the **buzzer module** like in the picture, so your robot buzzes SOS instead.
- If you're feeling really clever, put the LED module back into another output and add some more code so it flashes **and** buzzes the sequence for S.O.S!