

D - Transport the Nuclear Waste

We need to transport some very unstable nuclear waste across the planet, so we must program the robot to move as smoothly as we can.



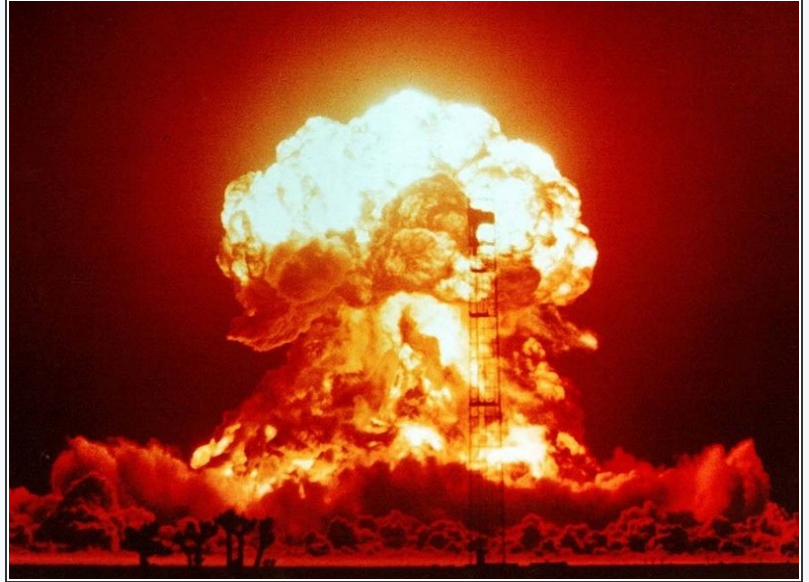
INTRODUCTION

We need to transport some very unstable nuclear waste across the planet, so we must program the robot to move as smoothly as we can.

Step 1

Nuclear Waste

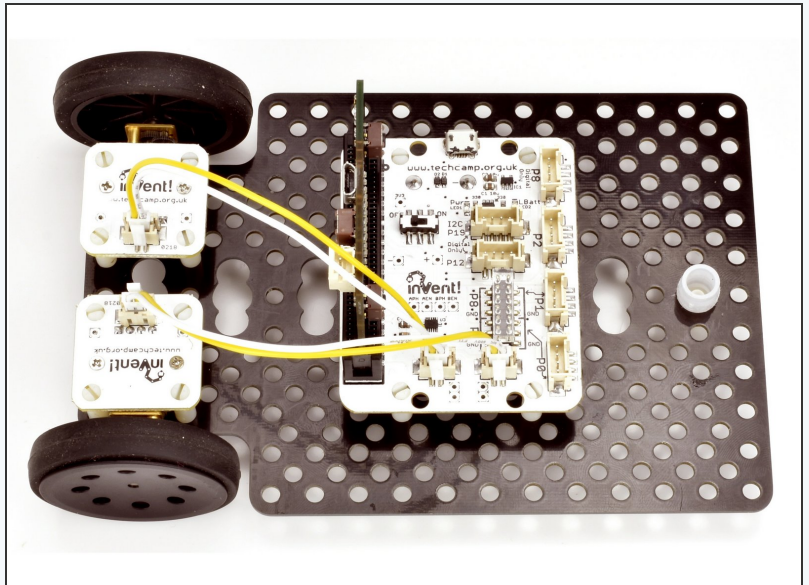
- Some **nuclear waste** has been found near the base, and we need to move it to the other side of the planet as it is **very dangerous**.
- The nuclear waste is **extremely unstable**, so we need to make our robot **accelerate** and **decelerate smoothly** so it doesn't explode!
- To do this, we are going to learn about **variables**.



Step 2

Assemble the Robot

- Put your robot together just like the picture! The connections should be:
- Left Motor > **M1**
- Right Motor > **M2**



Step 3

Test Program

- **Build** the program in the picture!
- **Before** you try programming your robot, **what do you think this program will do?**

```
9      p[6]=(p[5]+p[3])/2;p[7]=(p[4]+p[2])/2
10     while(p0()>p[7]and p1()>p[6]):d(0.1)
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     speed=100
17     drive_motor(0,speed)
18     sleep(1000)
19     drive_motor(0,0)
20
21
22
23
24
```

Step 4

What are Variables?

- A good way to understand variables is to think of them like your **lockers** at school.
- To use a locker, you need to put your **name** on it - we do the same with a variable, and you can call it **anything you want!**
- We can then put **whatever we like** inside the locker - books, bags, clothes, anything! We can do the same with variables, but for now we'll just put **numbers** inside them.
- We can **add, remove and change things** in the locker whenever we like - its the same for the number in the variable!
- Most usefully, we can go back to the locker or variable at any time and **see what's in it** (so long as we know the **name** of the locker or variable!).



Step 5

Using Variables

- If you don't quite understand, don't worry - for now, just remember we can do these things with variables:
 - **Call** them anything we like (variable **name**)
 - **Store** any number we like inside them (variable **contents**)
 - **Change** the contents at any time (add, subtract, multiply, divide and so on)
 - **Access** the contents at any time, so long as we know the **name** of the variable.
- To make a new variable in Python, you simply type its **name** (it can be whatever you want), followed by an equals sign, followed by the **value** you want to set it to.

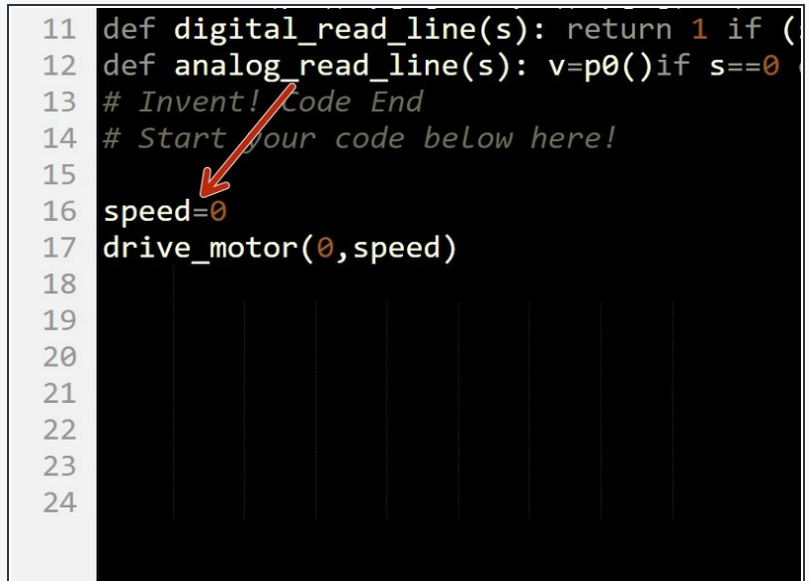


```
speed=100
```

Step 6

Accessing Variables

- Starting with a new program, make a new variable called **speed** and set it to **0**.
- **Access** the speed variable by setting both motors running using the **speed** variable.
- Notice how we only use **1 equals sign** when we make the variable and not 2 like in an if statement?
 - Using 1 equals sign **sets** one thing equal to another.
 - Using 2 equals signs **checks** if 1 thing is equal to another.
 - It is **very important** to always use the right number of equals signs!



```
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 speed=0
17 drive_motor(0,speed)
18
19
20
21
22
23
24
```

Step 7

Changing Variables

- Now let's try **changing** the motor speeds by **changing the variable**!
- First, **change** the line that makes the speed variable, so speed is set to **50**.
- Then, add a **sleep** of 1 second after the drive_motor line so the robot moves forwards for a bit at the first speed.
- Then, add a line that says: **speed=100**
- Now add another drive_motor line to set the motors running at **speed** again - but now speed has changed, so the motors should **change speed too**!

```
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 speed=50
17 drive_motor(0,speed)
18 sleep(1000)
19 speed=100
20 drive_motor(0,speed)
21
22
23
24
25
```

Step 8

Why do we need variables anyway?


```
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
17 drive_motor(0,50)
18 sleep(1000)
19 drive_motor(0,100)
20
21
22
23
24
25
```

```
17 drive_motor(0,0)
18 sleep(1000)
19 drive_motor(0,10)
20 sleep(1000)
21 drive_motor(0,20)
22 sleep(1000)
23 drive_motor(0,30)
24 sleep(1000)
25 drive_motor(0,40)
26 sleep(1000)
27 drive_motor(0,50)
28 sleep(1000)
29 drive_motor(0,60)
30 sleep(1000)
31 drive_motor(0,70)
32 sleep(1000)
33 drive_motor(0,80)
34 sleep(1000)
35 drive_motor(0,90)
36 sleep(1000)
37 drive_motor(0,100)
38
```

- You might be thinking - **why bother** using a variable to do this? We could have just used the **simple** program in the picture!
- Well, what if we wanted to increase the speed of our robot (**accelerate**) slowly?
- Even if we started the speed at **0** and increased the speed by just **10 every second** (0,10,20,30,40.....), the program would require **10 motor blocks!**
- Have a look at the second picture to see this program - it is **far too long**.

Step 9

Variables and Loops

- By combining **variables** and **loops**, we can program things like acceleration **very easily**.
 - **Have a look** at the program in the picture - can you **work out** what is going on?
 - The **range** function in the for loop has an optional **third number**: how much to increase the iterable by each time. In this case, we start from 0 and increase speed by **10** each time the loop runs.
 - So the for loop runs **10 times** - each time it runs, **speed is increased by 10**, and the speed of the motors is **set to speed**!
 - See how much **shorter** this program is?
 - **Write this program** and test it out.
-  If you can't remember how **for loops** work, have a look back at the **previous section**.

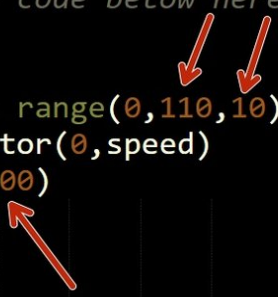
```
9      p[6]=(p[5]+p[3])/2;p[7]=(p[4]+p[2])
10      while(p0()>p[7]and p1()>p[6]):d(0 i
11  def digital_read_line(s): return 1 if (
12  def analog_read_line(s): v=p0()if s==0
13  # Invent! Code End
14  # Start your code below here!
15
16
17  for speed in range(0,110,10):
18      drive_motor(0,speed)
19      sleep(1000)
20
21
22
23
24
```

Step 10

Smooth Acceleration

- We're nearly there! Let's **change some things** in our program to make the robot accelerate **really smoothly**, so we don't set off the nuclear waste.
- Change the program so that:
 - The loop repeats every **0.1 seconds** (100 milliseconds) instead of every second
 - Each time the loop repeats, speed is **increased by 1**
 - The loop runs enough times for the motors to change speed from **0** all the way to **100**.
- If you need some **hints**, we've marked the parts of the program you will need to **change**!
- **Test your program** - your robot should now speed up really smoothly.

```
8 while(running_time() < 4000): v=[p
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 if
11  def digital_read_line(s): return 1 if (s
12  def analog_read_line(s): v=p0()if s==0 e
13  # Invent! Code End
14  # Start your code below here!
15
16
17  for speed in range(0,110,10):
18      drive_motor(0,speed)
19      sleep(1000)
20
21
22
23
24
```



Challenge - Smooth Deceleration

Challenge!



```

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
17  for speed in range(0,101):
18      drive_motor(0,speed)
19      sleep(100)
20
21
22
23

```

- We need to be able to **decelerate** (slow down) smoothly as well to stop on the other side of the planet.
- **Change your code** so that the motors start at 100, and **decelerate smoothly to 0**.
- If you need it, the second picture has the correct program for smooth **acceleration**.

Step 12

Transport the Waste

- Now you have learnt everything you need to **safely** move the waste!
- Write a program that:
 - **Starts** at your base
 - **Accelerates smoothly** towards the other side of the planet
 - **Decelerates smoothly** and **stops** at the other side
 - **Waits** for **5 seconds** so the waste can be unloaded
 - **Spins** on the spot **180** degrees
 - Drives back to base at **full speed**, and stops in the right place.

