Procedures

Introduction

You will aim to learn the following objectives and keywords during this lesson.

Learning Objective	- To identify a procedure - To demonstrate the use of procedure
Keywords	- Procedure - Functions - Comments

Setting the Scene

Concept	How it is used
A ***procedure*** is a series of steps that are stored to be called upon within the main code. A ***procedure*** performs a task where as a function returns data	When you write code, it can save time if you have a set of instructions that can be used whenever it is called upon. A ***procedure*** is a set of instructions that can be actioned within the main code by the name given to it.

Can you think of a time where you have learnt what to do and when you went to do the same task again, you did not need instructions to complete it?

You can create many procedures alongside the main code with a name given to it to call upon in the main code. Within Blockly, both are created using the block ‘function’.

Activity 1

Activity 1 is focused on getting students to think about what a ***procedure*** is and relate this to their everyday tasks.

Scenario 1	Scenario 2
You are making a cake and the instructions ask you to add flour and mix with margarine and egg. There are separate procedures for: - How to add enough flour needed? - How to add margarine needed? - How to crack an egg and add it to the mix	You want to play on your Xbox, how do you know what to do when you log on? - What is the log on process? - How do you access a game? - How do you use the controller within the game?

Can you think of 2 other procedures you might follow in a normal day?

Small Group Activity

In pairs, or groups of 3 look at the list of tasks in front of you. Split them into two groups: 1. Procedures, 2. Algorithm Steps

Procedures	Algorithm Steps

As humans you learn from the tasks you complete, when a task is learnt, like how to read and write, the knowledge is stored to be called upon when you next need it to read a book or write a letter.

Additional Small Activity

To create a function in Python it needs to be defined at the start of the code.

You need to use ‘`def`’ followed by the name given to the function in this case ‘`aFunction`’ and a pair of brackets. All the code to be placed within the function is indented within it.

Put at the top of your code program, this looks like this:

```
def aFunction():
     #code goes here
```

The function is then called upon within the main code whenever the following code is added by writing the function name, in this case ‘`aFunction`’:

```
aFunction()
```

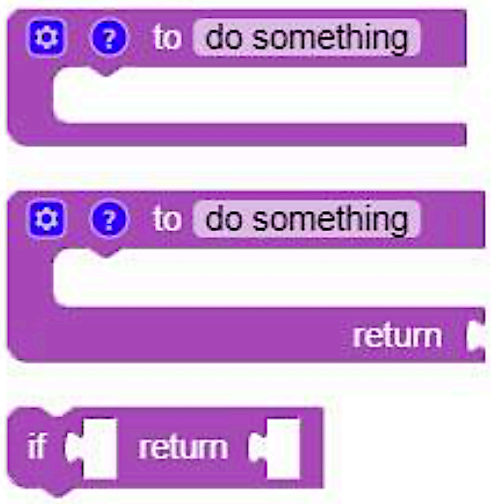
Below is the code for creating a function to output the words ‘Welcome MiRo’:

```
def aFunction():
    print(“Welcome MiRo”)
```

What would the output be in the following code?	
Code	Output

def aFunction():
    print(“Hello MiRo”)
 	
aFunction()
aFunction()	

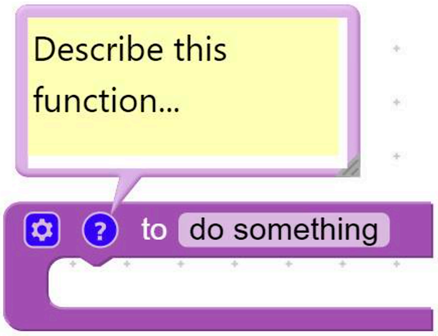
In computing terms ***functions*** and ***procedures*** are different as functions return a singles value where procedures hold a series of tasks. In block code the function blocks can be either.



What are the blocks available for *procedures*? They are located within the ‘**functions**’ tab as in block based code the function blocks can be created to represent a procedure or a function.

To name a function or procedure in block based code you edit the text from the default text ‘*do something*’.

It is good practice when writing any code, to add ***comments*** where possible to remind yourself what the code is to do or to allow other people to collaborate on a coding project. They can then see the description or notes on the parts of code completed.



Click the blue question mark to access the comments area to describe the function you are creating.

Within python programming a comment is written with a ‘#’ before it to define it as a comment and not part of the program.

Why would commenting within code help a programmer or a project?

Activity 2

For this activity you will be on your computer using MiRoCODE. Remember to save your work for further review!

How can we get MiRo to use a ***procedure*** to facilitate facial movements? We are going to **create a program to define the procedures to blink**.

What steps do you need to consider for creating a blink? How might this be written out?

Can you write some instructions out for the algorithm to blink below, which we will make into a procedure?

Blink: ....

....

....

....

....

What would the name of your procedure or function be?

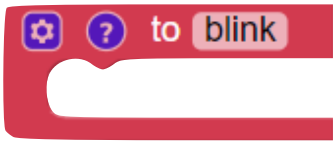
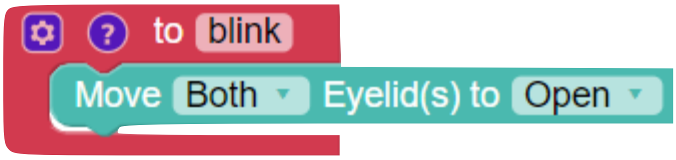
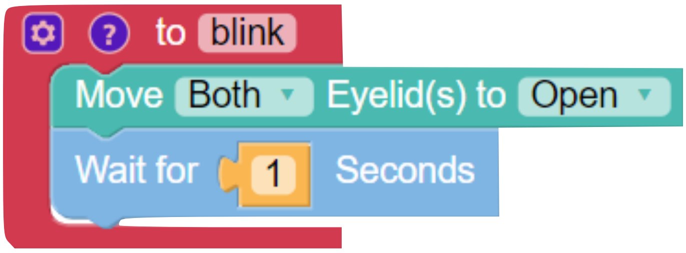
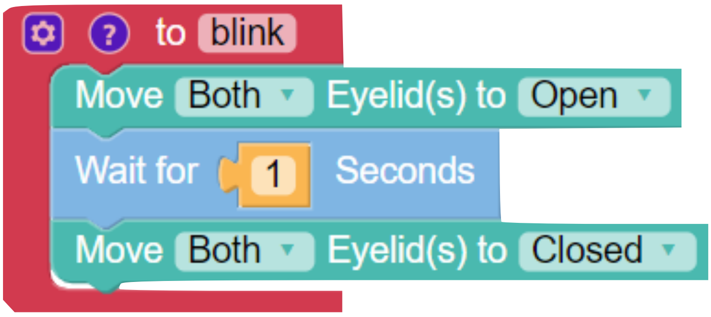
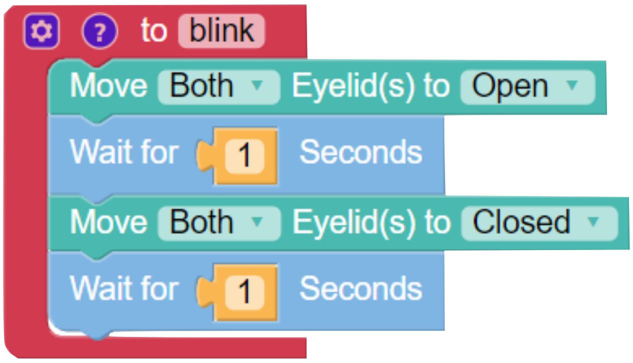
Why is the name of the procedure just as important as the content?

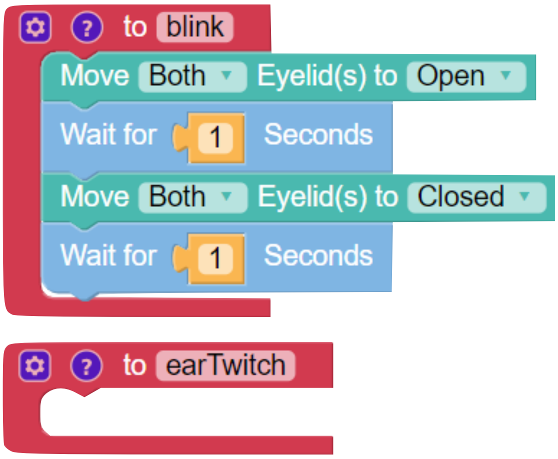
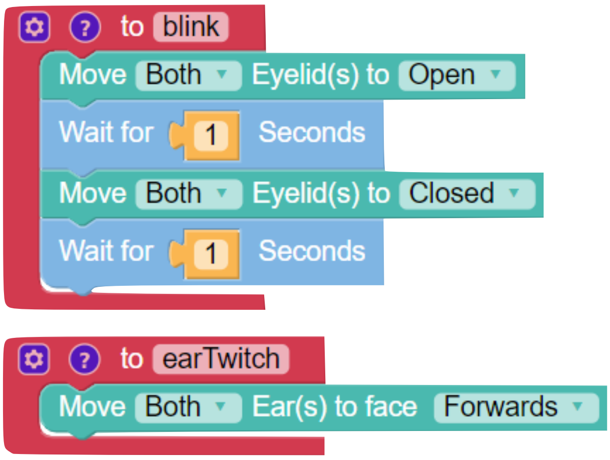
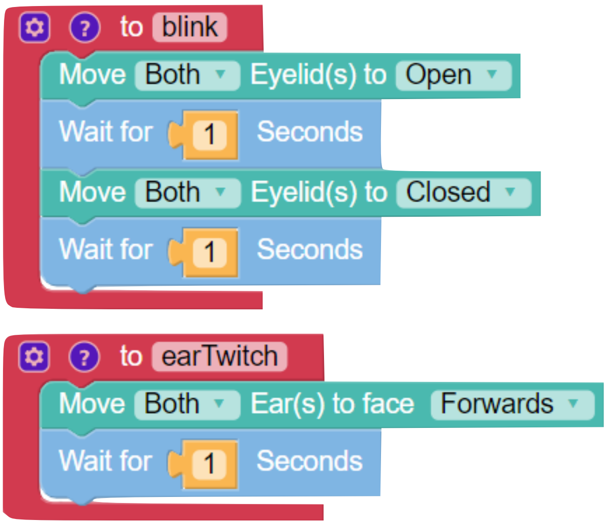
Using this algorithm as your plan, create and run the program in the MiRoSIM.

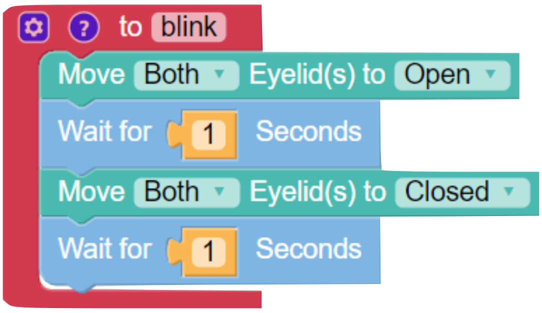
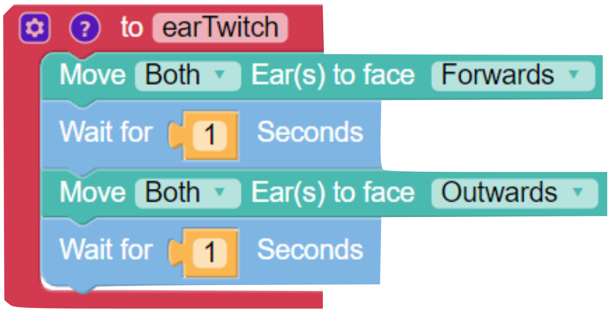
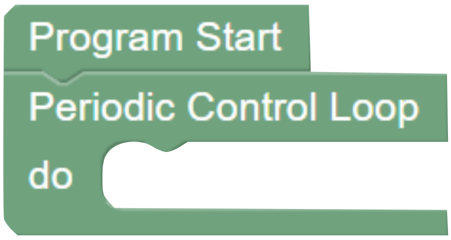
Summary Self-Assessment

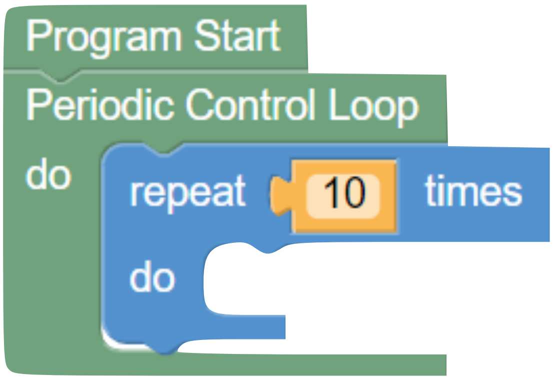
Question	Got it		Got it with help		Unsure	
Can you identify a procedure?	□		□		□	
Can you describe what a procedure is?	□		□		□	
Can you create and use a procedure in the MiRoCODE?	□		□		□	

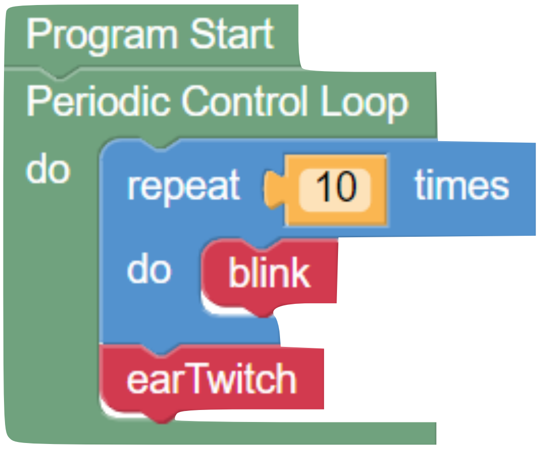
PART 1 Step-by-Step - Create two functions to control MiRo's blinking and ear twitching.

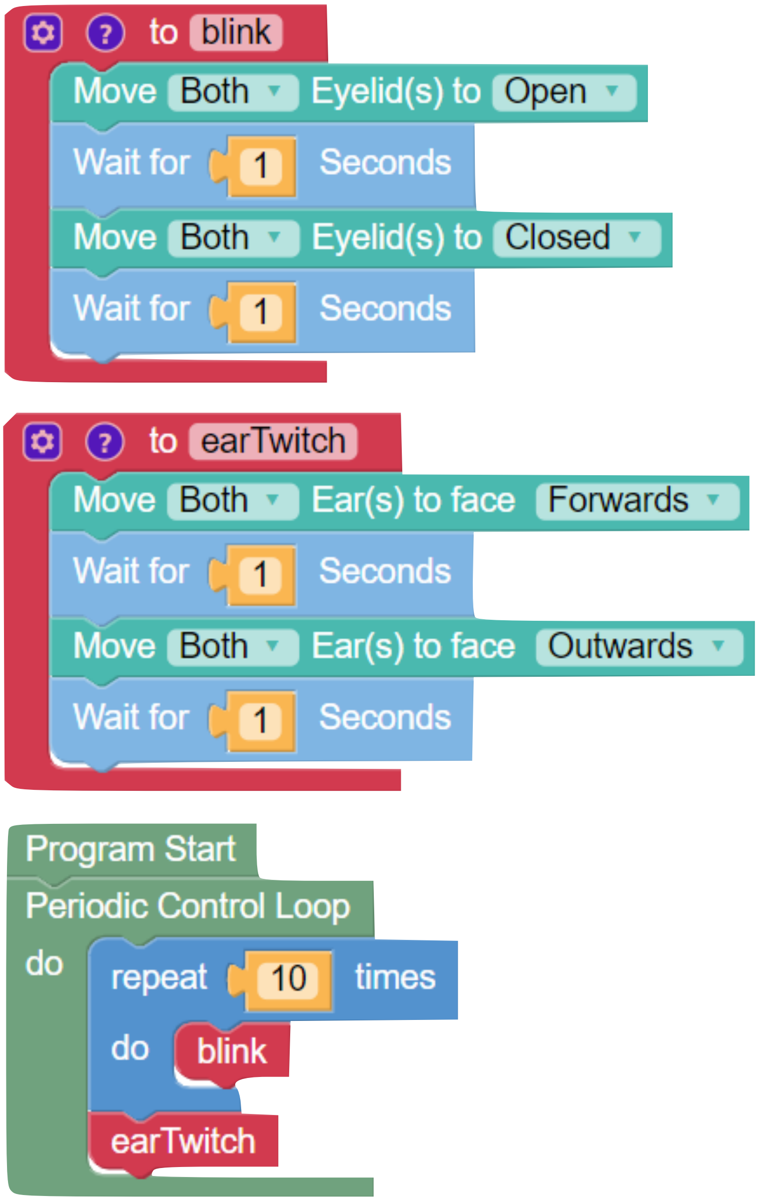
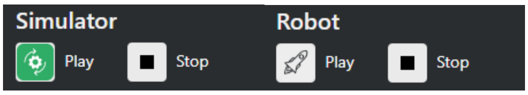
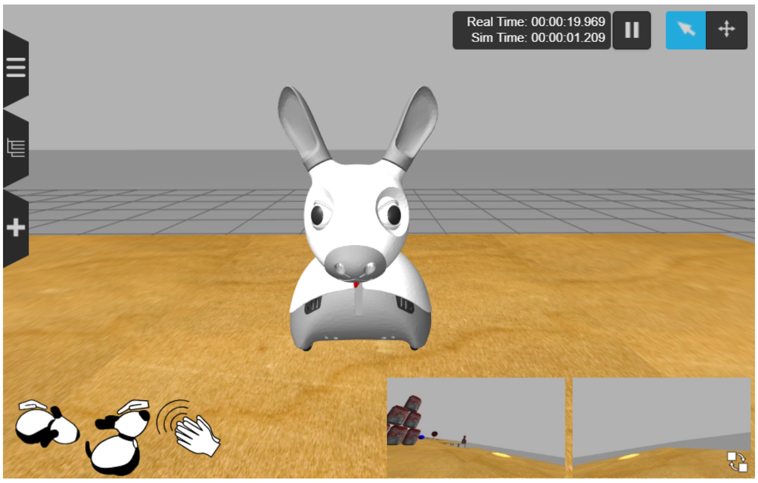
	Step	Cat.	Image
1	Add **'to do something'** from **'Functions'**		 to blink
2	Set the function name to 'blink'. Add a comment to state it is a function to make MiRo blink.		
3	Add **'move left eyelid(s) to open'** block from **'Simple Motion'.**		 to blink Move Both Eyelid(s) to Open
4	Connect into 'blink' function. Set to both eyes.		
5	Add '**wait for 1 seconds'** block from '**Control'**. Connect to 'move both Eyelid(s) to Open' block.		 to blink Move Both Eyelid(s) to Open Wait for 1 Seconds
6	Connect to 'move both Eyelid(s) to Open' block.		 to blink Move Both Eyelid(s) to Open Wait for 1 Seconds Move Both Eyelid(s) to Closed
7	Connect to 'wait for 1 seconds' block Set to both eyes and closed. **** You can right click and duplicate blocks already on workspace.**		
8	Add '**wait for 1 seconds**' block from '**Control'**. Connect to 'move both Eyelid(s) to Open' block.		 to blink Move Both Eyelid(s) to Open Wait for 1 Seconds Move Both Eyelid(s) to Closed Wait for 1 Seconds

Step		Cat.	Image
9	Add **‘to do something’** from **‘Functions’**		 to blink Move Both Eyelid(s) to Open Wait for 1 Seconds Move Both Eyelid(s) to Closed Wait for 1 Seconds to earTwitch
10	Rename the function to ‘earTwitch’ ** **Remember to add a comment by pressing the question mark.**		
11	Add **‘move left ear(s) to face forwards’** block from **‘Simple Motion’.**		 to blink Move Both Eyelid(s) to Open Wait for 1 Seconds Move Both Eyelid(s) to Closed Wait for 1 Seconds to earTwitch Move Both Ear(s) to face Forwards
12	Connect into ‘earTwitch’ function. Set to both ears.		
13	Add ‘**wait for 1 seconds’** block from ‘**Control**’. Connect to ‘move both Eyelid(s) to Open’ block.		 to blink Move Both Eyelid(s) to Open Wait for 1 Seconds Move Both Eyelid(s) to Closed Wait for 1 Seconds to earTwitch Move Both Ear(s) to face Forwards Wait for 1 Seconds
14	Connect to ‘move both Ear(s) to face forwards’ block. ** **You can right click and duplicate blocks already on workspace.**		

Step		Cat.	Image
15	Duplicate both blocks within the function **'earTwitch'** Connect in same order to '**wait for 1 seconds'** block'.		 to blink Move Both Eyelid(s) to Open Wait for 1 Seconds Move Both Eyelid(s) to Closed Wait for 1 Seconds  to earTwitch Move Both Ear(s) to face Forwards Wait for 1 Seconds Move Both Ear(s) to face Outwards Wait for 1 Seconds
16	Set to both ears to face outwards.		
17	Add '**Program Start**' from '**Setup**' and add '**Periodic Control Loop do**' and connect.		 Program Start Periodic Control Loop do

18	Add '**repeat 10 times**' from '**Loops**' and connect into the loop.		
			Program Start
			Periodic Control Loop
			do
			repeat 10 times
			do

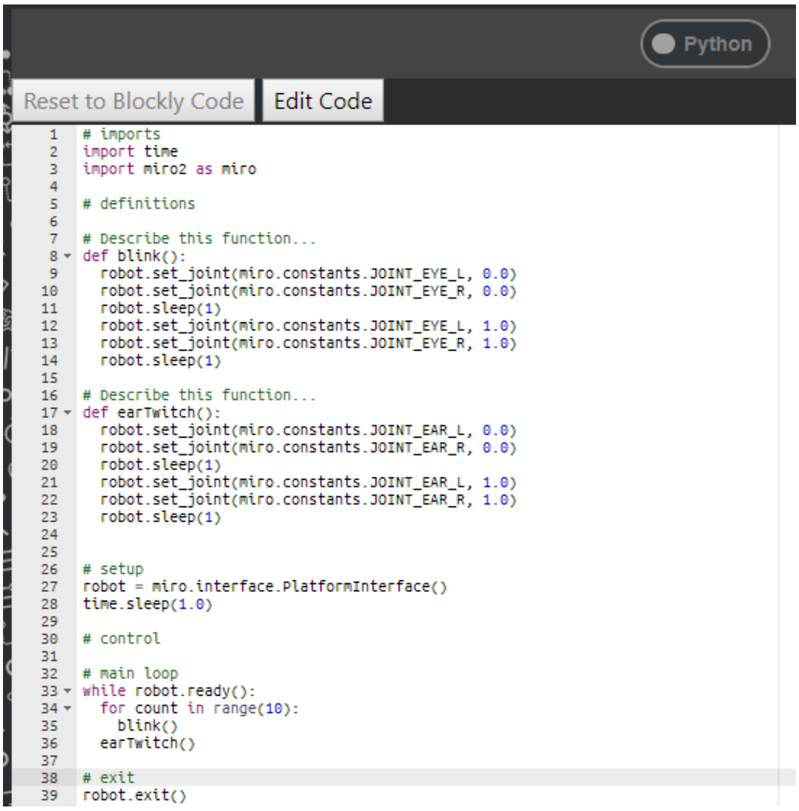
19	Add '**blink**' from '**Functions**', connect inside the 'repeat' loop. Add '**earTwitch**' from '**Functions**', connect in after the 'repeat 10 times' block.		 Program Start Periodic Control Loop do repeat 10 times do blink earTwitch

	Step	Image
20	Duplicate both blocks within the function **‘earTwitch’** Connect in same order to ‘**wait for 1 seconds’** block’.	 to blink Move Both Eyelid(s) to Open Wait for 1 Seconds Move Both Eyelid(s) to Closed Wait for 1 Seconds to earTwitch Move Both Ear(s) to face Forwards Wait for 1 Seconds Move Both Ear(s) to face Outwards Wait for 1 Seconds Program Start Periodic Control Loop do repeat 10 times do blink earTwitch
21	Click **Robot Play** OR **Simulator Play**	
	Remember You will need to zoom in on MiRo’s face if you are using the simulator.	

PART 2 Step-by-Step - Moving into Python

Step | **Image**

1 Flip to Python code by clicking on the ‘Blockly’ button.



```
# imports
import time
import miro2 as miro

# definitions

# Describe this function...
def blink():
  robot.set_joint(miro.constants.JOINT_EYE_L, 0.0)
  robot.set_joint(miro.constants.JOINT_EYE_R, 0.0)
  robot.sleep(1)
  robot.set_joint(miro.constants.JOINT_EYE_L, 1.0)
  robot.set_joint(miro.constants.JOINT_EYE_R, 1.0)
  robot.sleep(1)

# Describe this function...
def earTwitch():
  robot.set_joint(miro.constants.JOINT_EAR_L, 0.0)
  robot.set_joint(miro.constants.JOINT_EAR_R, 0.0)
  robot.sleep(1)
  robot.set_joint(miro.constants.JOINT_EAR_L, 1.0)
  robot.set_joint(miro.constants.JOINT_EAR_R, 1.0)
  robot.sleep(1)


# setup
robot = miro.interface.PlatformInterface()
time.sleep(1.0)

# control

# main loop
while robot.ready():
  for count in range(10):
    blink()
    earTwitch()

# exit
robot.exit()
```

Can you find the two functions?

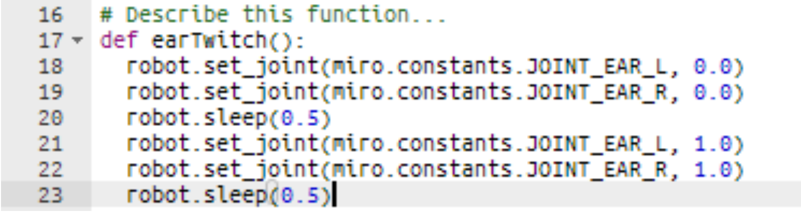
What code defines a function?

2 Locate the lines in each function that control the time between actions.

```
robot.sleep(1)
```

3 Edit the time set within the code from ‘1’ to ‘0.5’.

There are 4 locations to edit, 2 in each function.



```
# Describe this function...
def earTwitch():
  robot.set_joint(miro.constants.JOINT_EAR_L, 0.0)
  robot.set_joint(miro.constants.JOINT_EAR_R, 0.0)
  robot.sleep(0.5)
  robot.set_joint(miro.constants.JOINT_EAR_L, 1.0)
  robot.set_joint(miro.constants.JOINT_EAR_R, 1.0)
  robot.sleep(0.5)
```

4 **Run your program**

Experiment with the settings and investigate:

Change the time further up and down to observe the change in output.