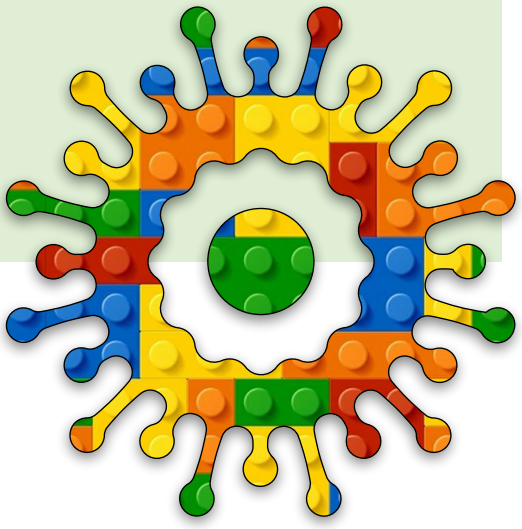
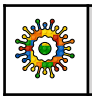
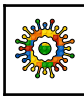
Introducing Robotics Module A Unit #3 Functions





Contents

Module A Unit #3 Functions



Introduction to functions

Segmenting code into functions allows a programmer to create modular pieces of code that perform a defined task and then return to the area of code from which the function was "called". The typical case for creating a function is when one needs to perform the same action multiple times in a programme.

Sketch A3.1 a blinking function

We have a function called `void setup()`, and a function called `void loop()`. Now we are going to create one called `void blink()`. This is a made-up function where we are going to put our code for blinking.

```
int delayPeriod = 250;

void setup()
{
  pinMode(11, OUTPUT);
}

void loop()
{
  for (int i = 0; i < 10; i++)
  {
    digitalWrite(11, HIGH);
    delay(delayPeriod);
    digitalWrite(11, LOW);
    delay(delayPeriod);
  }
  delay(2000);
}

void blink()
{

}
```

🗒️ Notes

We can put the function anywhere, but it does start at the beginning and work its way down. The function doesn't do anything yet; it is just an empty shell.

🛠️ Code Explanation

`void blink()`	We have created a new function called blink()

Sketch A3.2 calling the blink

Now we are going to call it from inside the `for()` loop. This is a cut-and-paste job, taking the code out of the for() loop and replacing it with the `blink()` function.

```
int delayPeriod = 250;

void setup()
{
  pinMode(11, OUTPUT);
}

void loop()
{
  for (int i = 0; i < 10; i++)
  {
    blink();
  }
  delay(2000);
}

void blink()
{
  digitalWrite(11, HIGH);
  delay(delayPeriod);
  digitalWrite(11, LOW);
  delay(delayPeriod);
}
```

🗒️ Notes

This can be useful for keeping the code neat and tidy, especially where you want to use the same bit of code repeatedly. One of the targets that coders try to achieve is to write as few lines as possible to achieve the same result. It is considered bad form to repeat lines of code unnecessarily.

🛠️ Code Explanation

`blink();`	This blink() function is called ten times in the for() loop

Sketch A3.3 is this an argument?

Here we will combine functions and arguments. Using the above sketch, we can rationalise it even more by using the two arguments in the function itself. We will start with just one and then add the other afterwards. We will also change the `delayPeriod` to `100`.

❗ Remove the line of code: `int delayPeriod = 250;`

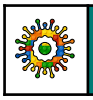
```
void setup()
{
  pinMode(11, OUTPUT);
}

void loop()
{
  for (int i = 0; i < 10; i++)
  {
    blink(100);
  }
  delay(2000);
}

void blink(int delayPeriod)
{
  digitalWrite(11, HIGH);
  delay(delayPeriod);
  digitalWrite(11, LOW);
  delay(delayPeriod);
}
```

Notes

You will notice that it blinks much faster (or should do).

🛠️ Code Explanation

`void blink(int delayPeriod)`	This is a variable inside a function, you can call it an argument. It will hold that value when it is called from somewhere else, in this case 100



Sketch A3.4 count to 10

We can go one step further by adding in the count value of 10. There is a bit of cutting and pasting, so work through the sketch to see how and why it is doing exactly what it does. We will change the rate of blinking so you can see a change.

Remove the `for()` loop from the `void loop()` and put it in the `void blink()`. If in doubt, just retype the whole thing; it is good practice anyway!

```
void setup()
{
  pinMode(11, OUTPUT);
}

void loop()
{
  blink(100, 10);
  delay(2000);
}

void blink(int delayPeriod, int count)
{
  for (int i = 0; i < count; i++)
  {
    digitalWrite(11, HIGH);
    delay(delayPeriod);
    digitalWrite(11, LOW);
    delay(delayPeriod);
  }
}
```

🗒️ Notes

You have initialised them inside the function brackets. You draw their values from the `void loop()` function.

🌻 Challenge

Change the values in `blink(100, 10)`.

🛠️ Code Explanation

`void blink(int delayPeriod, int count)`	Both are variables (`delayPeriod` and `count`) that are also arguments. They each take on the values of 100 and 10 respectively

Sketch A3.5 a blinking stop

Start a completely new sketch (or cut and paste). In this next sketch, we will use the += operator to slow the blink down. First, this is our starting sketch.

```
int delayPeriod = 1000;


void setup()
{
  pinMode(11, OUTPUT);
}

void loop()
{
  digitalWrite(11, HIGH);
  delay(delayPeriod);
  digitalWrite(11, LOW);
  delay(delayPeriod);
}
```

Sketch A3.6 increased blinking

Changing the starting delay period to something much shorter, 10 milliseconds, and increasing it by 20 milliseconds on each blink (iteration).

```
int delayPeriod = 10;

void setup()
{
  pinMode(11, OUTPUT);
}

void loop()
{
  digitalWrite(11, HIGH);
  delay(delayPeriod);
  digitalWrite(11, LOW);
  delay(delayPeriod);
  delayPeriod += 20;
}
```

🌻 Challenge

Try replacing `delayPeriod += 20;` with `delayPeriod *= 2;`

🛠️ Code Explanation

`delayPeriod += 20;`	We add 10 to the variable delayPeriod on every iteration

Sketch A3.7 stop blinking

If we wanted to stop the blinking at 1000 milliseconds, we would need to first introduce another variable. We will call this increment and make it much bigger, i.e. 50.

```
int delayPeriod = 10;
int increment = 50;

void setup()
{
  pinMode(11, OUTPUT);
}

void loop()
{
  digitalWrite(11, HIGH);
  delay(delayPeriod);
  digitalWrite(11, LOW);
  delay(delayPeriod);
  delayPeriod += increment;
}
```

Sketch A3.8 constantly blinking

Stopping at 1 second (1000 milliseconds), so that when the delay is one second, it then blinks at a constant rate.

```
int delayPeriod = 10;
int increment = 50;

void setup()
{
  pinMode(11, OUTPUT);
}

void loop()
{
  digitalWrite(11, HIGH);
  delay(delayPeriod);
  digitalWrite(11, LOW);
  delay(delayPeriod);
  delayPeriod += increment;
  if (delayPeriod > 1000)
  {
    increment = 0;
  }
}
```

Challenge

Try the following:

1. Replace `increment = 0;` with `delayPeriod = 0;`
2. Replace `increment = 0;` with `increment = -increment;`