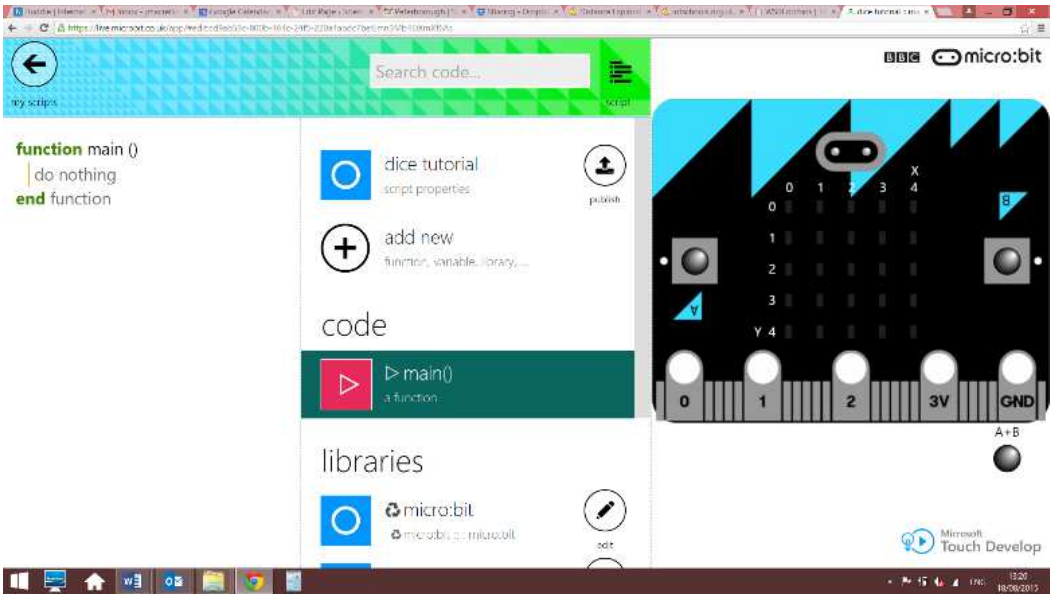
INSTRUCTION GUIDE 1: HOW TO MAKE A MICRO:BIT DICE

This is a very simple way to start programming your micro:bit using code which is easy to understand. The following outline is one of several ways to do this, which is particularly handy as it allows you to easily add more numbers, should you wish to have a 12-sided dice as opposed to 6-sided, for example.

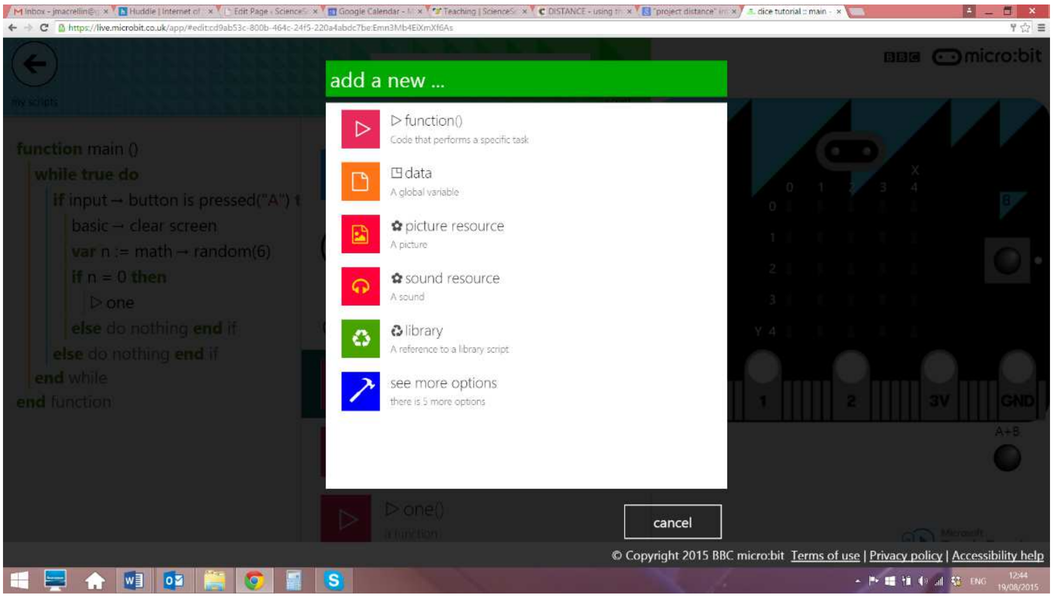
Creating your functions

Go to 'script' and click 'add new':



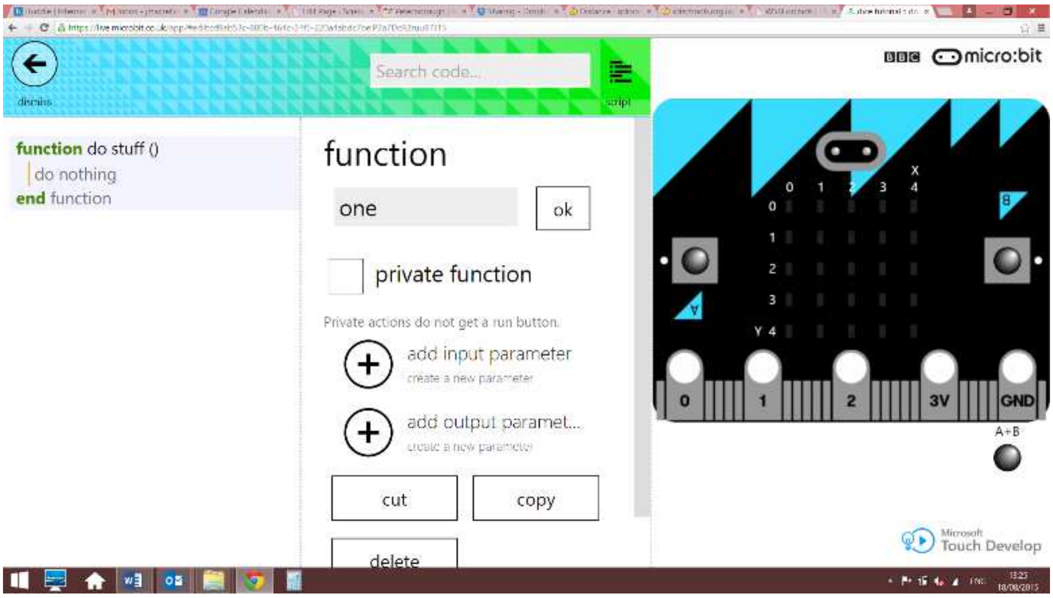
```
function main ()
    do nothing
end function
```

Select 'function' from the menu:



```
function main ()
    while true do
        if input → button is pressed("A") t
            basic → clear screen
            var n := math → random(6)
            if n = 0 then
                ▷ one
            else do nothing end if
        else do nothing end if
    end while
end function
```

Click on the title of your newly created function to rename it. For a six-sided dice, generate functions called one, two, three, etc. up to six. There is no need to add any input or output parameters.

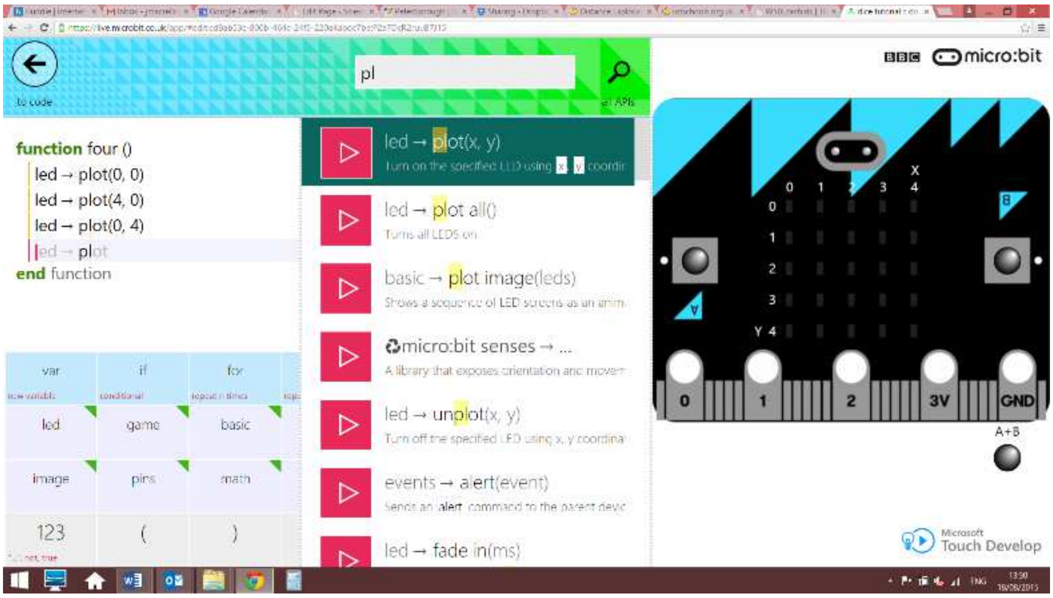


```
function do stuff ()
    do nothing
end function
```

Under each function, enter commands to plot the LEDs by typing plot and selecting the plot function from the menu.

Plot the LEDs as shown on a normal dice. For example, for the number four, enter the following commands:

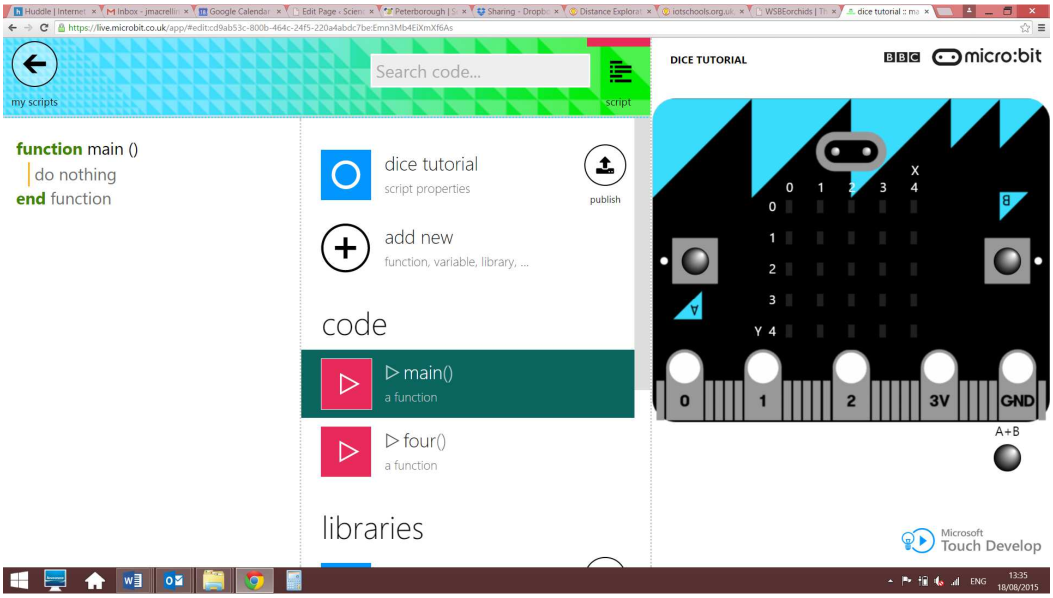
```
led -> plot (0, 0)
led -> plot (4, 0)
led -> plot (0, 4)
led -> plot (4, 4)
```



```
function four ()
    led → plot(0, 0)
    led → plot(4, 0)
    led → plot(0, 4)
    led → plot
end function
```

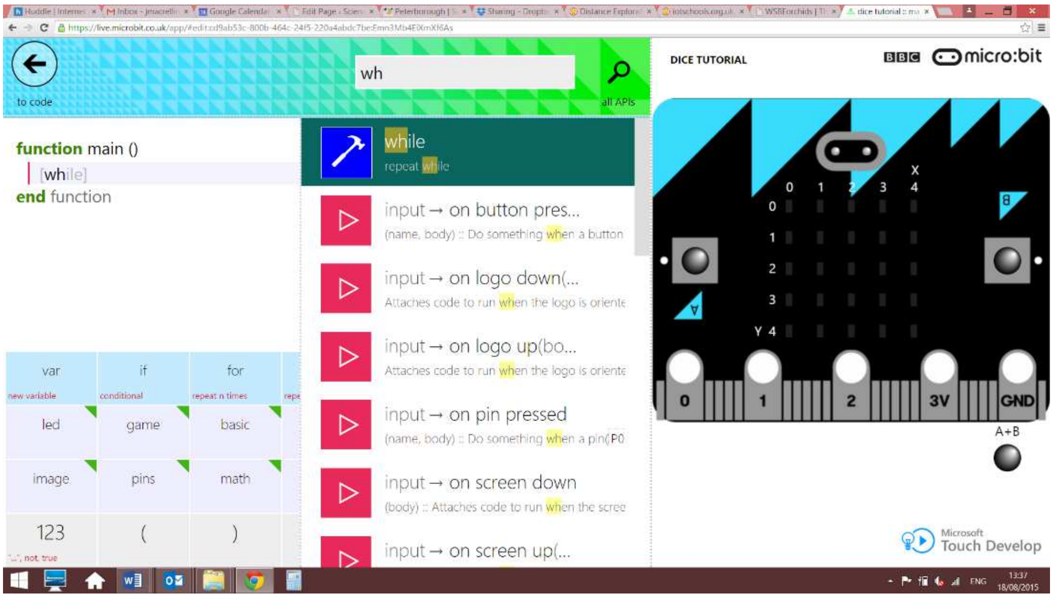
Creating random numbers

Once you have created your functions open the ‘script menu’ and return to the main code by clicking on the ‘main’ option.



```
function main ()
    do nothing
end function
```

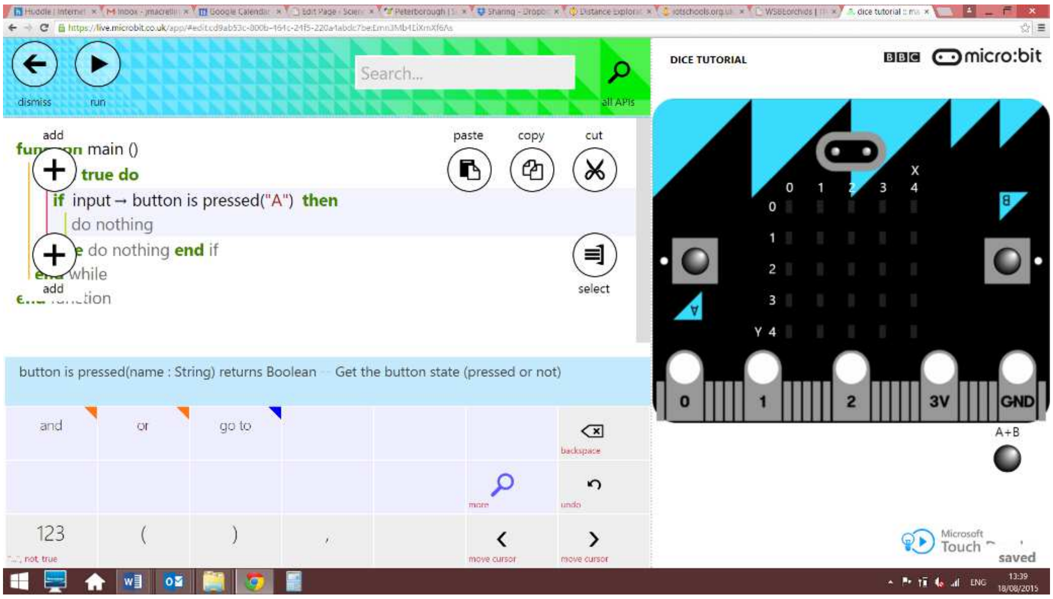
Under ‘function main’ create a loop by typing in ‘while’ and selecting ‘while’ from the menu:



```
function main ()
    [while]
end function
```

Add an ‘if’ statement within the loop so that the code will only execute when the button is pressed:

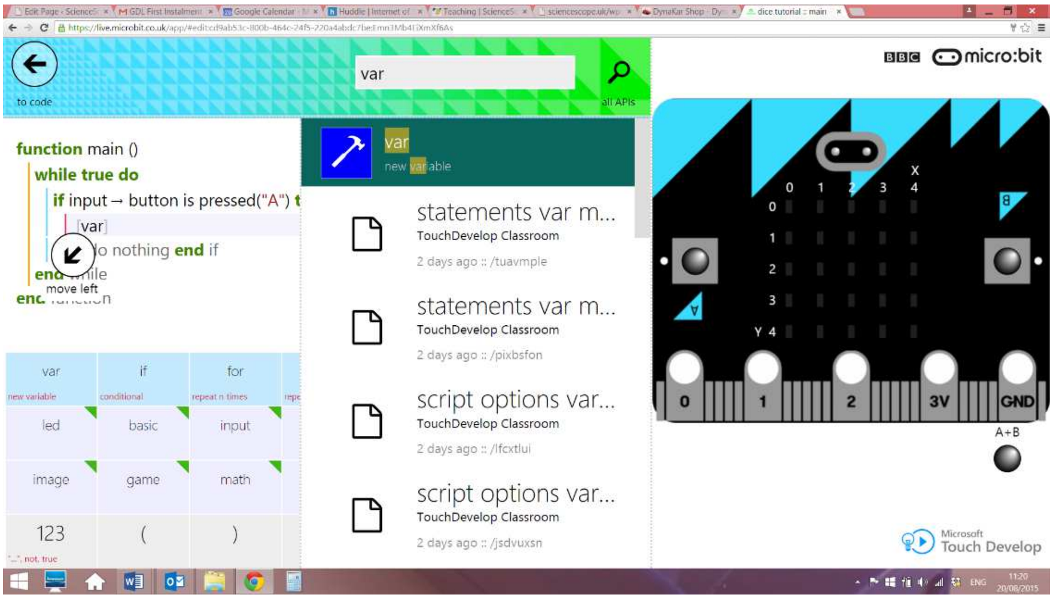
if input -> button is pressed(“A”)



```
function main ()
  while true do
    if input → button is pressed("A") then
      do nothing
    else do nothing end if
  end while
end function
```

button is pressed(name : String) returns Boolean   Get the button state (pressed or not)

The micro:bit has an inbuilt random number generator function which you can use to select your functions at random like a dice. To do this you need create a new variable within the ‘if’ statement. Click underneath the ‘if’ statement, type ‘var’ and select the ‘new variable’ command from the menu.



```
function main ()
  while true do
    if input → button is pressed("A") t
      [var]
    do nothing end if
  end while
end function
```

var
new variable

statements var m...
TouchDevelop Classroom
2 days ago :: /tuavmple

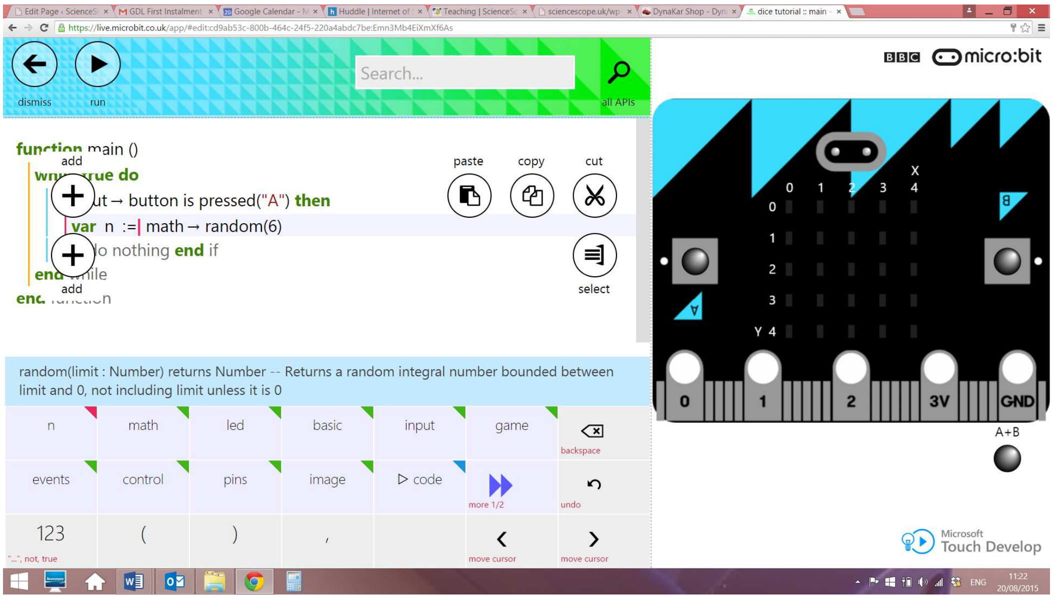
statements var m...
TouchDevelop Classroom
2 days ago :: /pixbsfon

script options var...
TouchDevelop Classroom
2 days ago :: /lfcxtlui

script options var...
TouchDevelop Classroom
2 days ago :: /jsdvuxsn

Call your variable 'n'; 'n' will select which 'side' of the dice to show on the screen. There are six sides so you need 'n' to select from a choice of six numbers. You can make it do this by calling the built-in random number function as follows:

n := math > random(6)



```
function main ()
  wh   rue do
    ut→ button is pressed("A") then
      var n := math→ random(6)
    o nothing end if
  end    ile
end   ion
```

random(limit : Number) returns Number -- Returns a random integral number bounded between limit and 0, not including limit unless it is 0

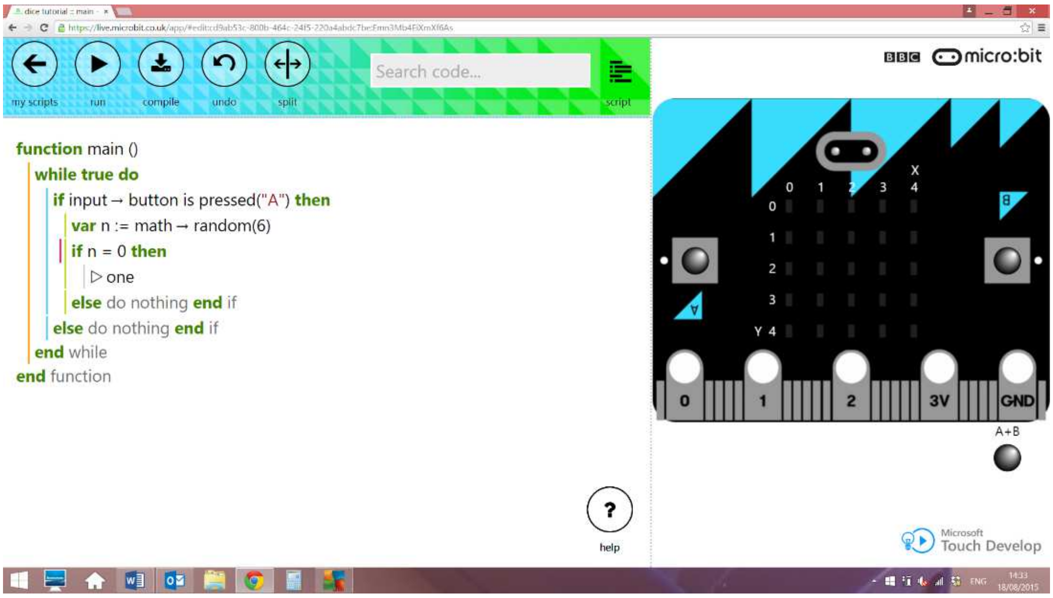
NB: the random function works with numbers from zero up to, but not including, the number you put in the brackets, so this will give you six different numbers, from zero to five.

Calling the functions

You are now ready to call the functions depending on the value of ‘n’, which you can do by adding an ‘if-else’ statement after the random number has been selected as follows:

if n = 0, call function one

else if n = l, call function two etc.



```
function main ()
  while true do
    if input → button is pressed("A") then
      var n := math → random(6)
      if n = 0 then
        ▷ one
      else do nothing end if
    else do nothing end if
  end while
end function
```

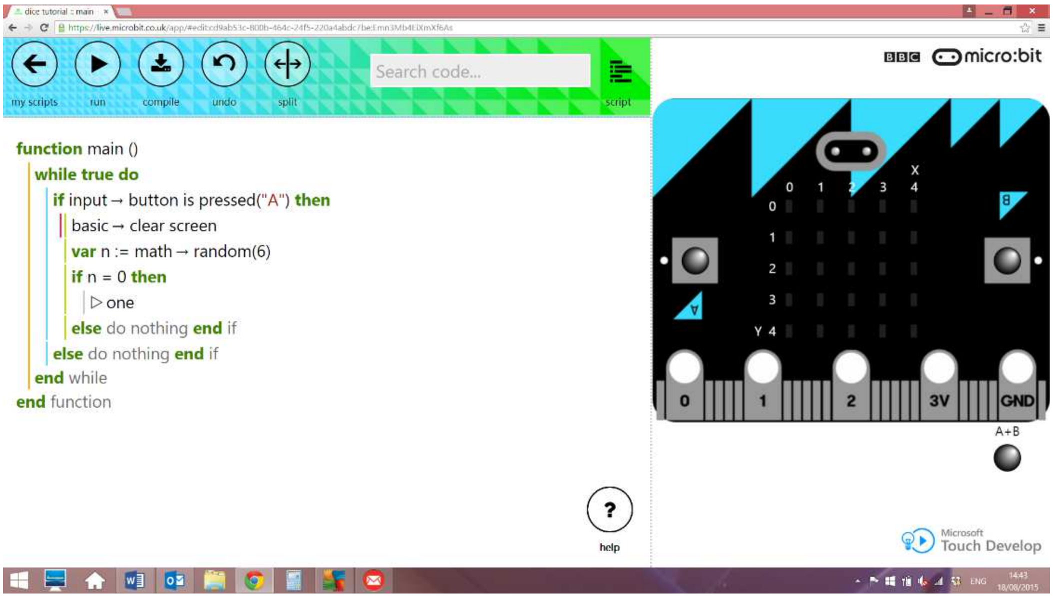
It is a good idea to add a small delay after the ‘if-else’ statement so that it will hold on the rolled number before repeating the code and generating a new one.

You can do this by entering the command pause and selecting ‘basic -> pause’ from the drop down menu. Enter the number of milliseconds you want for the delay into the brackets. So for a delay of 1 second your code should look like this:

basic -> pause (1000)

You also need to add a clear screen function otherwise the LEDs from the previous roll will remain on in addition to the ones from the new roll. You can do this by typing the command ‘clear screen’ and selecting ‘basic -> clear screen’ from the drop down menu.

This needs to be added within the button press statement but before the number is output onto the screen:



```
function main ()
  while true do
    if input → button is pressed("A") then
      basic → clear screen
      var n := math → random(6)
      if n = 0 then
        ▷ one
      else do nothing end if
    else do nothing end if
  end while
end function
```

Congratulations you have coded your dice!