![Logo

Description automatically generated]()**Processing 4**

**Line Drawing**

Open **trinket.io** in the browser. You can **sign-up** with your own email, or **log-in** with my account [steve.battle@uwe.ac.uk](mailto:steve.battle@uwe.ac.uk), using the password code-club67

This is a shared account, so add your **name** to any files you save

1. Lines can be drawn with the **line** function. The first two *arguments* set the x,y location of the start of the line, and the last two set end of the line.
2. Create a new Python Trinket for this “line drawing” code. I’ve created some **global** variables for x0,y0 (the line start) and x1,y1 (the end of the line). This draws a single line from the **origin** (0,0) to 100,100.

Graphical user interface, text, application

Description automatically generated

1. Notice how we can string assignments of the same value together, like x0 = x1 = 0 (this is because x1 = 0 returns 0).
2. When you press the mouse button or click on the trackpad, the **mousePressed** function is called. We can use this to change the start of the line, x0,y0. Add this function:

Text

Description automatically generated

1. Whenever we **change** a global variable value inside a function, we have to let python know it’s global, so it doesn’t create another variable **local** to the function.
2. Drag the mouse (hold the button while moving the mouse) to set the end of the line, x1,y1. Add this function:

Text

Description automatically generated

1. Now you can draw a single line. You can still see the line after you’ve released the mouse button because the ends are stored in x0,x1,x1,y1. These get overwritten when you draw another line.
2. To keep all the lines we draw we have to store them in memory. We can save them in an array. Declare an array called lines with your other global variable declarations. An array is a block of memory, indicated by square brackets, that can be indexed with e.g. lines[0], lines[1],…

**lines = []**

1. When we release the mouse button, we save the line in the array. We can save all four coordinates together in a single **tuple**, *appending* them to the array.

Text

Description automatically generated

1. All we have to do now is draw the lines in the array. Add the following code to the end of the **draw** function. We use a **for** loop to loop around each of the stored lines. The variable, i, is the loop variable, going up by one each time we go round the loop. The loop ends when we reach the **len**gth of the lines array.



1. The array elements are accessed with lines[i].
2. The elements of the saved tuple are accessed by [0], [1], [2], [3]
3. **SAVE your code.**