A Review: What is Programming?

• What is programming exactly?
• What is the essence of programming?
• Who needs it?
• Why do we need to understand it better?
• Why are they called languages?
• What is the best programming language?
• Name a few programming languages and their applications?
• Most Importantly, what is your background in programming?
• Why do you want to learn more about programming?
What is Processing?

It is a computer programming language with a weird name.

It is based on a real language, Java, so you will learn some Java code today.

It is designed for artists and creative people so it is easier than regular Java.

Programs in processing are called *sketches*.

Processing is great to make programming interactive graphics easier such as images, animations, and interactions.
Processing Family Tree

Arduino
Wiring
Design By Numbers
PostScript

Processing

C
C++
Java
Android Processing

Processing.js
JavaScript
HTML 5
The Processing Environment

You write and run your programs from the Processing Environment (PDE)
The Processing Coordinate System

We use coordinates to draw things on the screen
We use the command in Processing called `point(x,y)`

So to draw a point at 4 across and 5 down we write: `point(4,5);`

*(all commands in Java and Processing must have a ; at the end)*

To try it yourself, start Processing on your computer by typing ./processing

Then type in:

`point(4,5);`

Now press the play button and see what happens. Can you see a little dot?

Can you draw the dot in a different spot by changing your program?
Just like artists, in Processing we draw on a canvas.  

We can make our canvas bigger by using the \texttt{size(x,y)} command

Add the following command so you now have:
\begin{verbatim}
size(50,50);
point(4,5);
\end{verbatim}

Try and make your canvas bigger. How big can you make it?
To draw a line we use the command `line(x1, y1, x2, y2)`

This draws a line joining the points \((x_1, y_1)\) and \((x_2, y_2)\). \(x_1, y_1, x_2, y_2\) are called variables. You choose what value they have.

Try the following commands:

```plaintext
size(200, 200);
line(10, 20, 10, 50);
```

This draws a line from 10 across, 20 down to 10 across, 50 down.
Sometimes you want to provide notes about your program so you want the computer to ignore the text on that line.

If you put // in front of your commands the computer will ignore them. These are called “comments”

Try:
```java
size(200,200);
//line(10,20,10,50);
line(30,20,30,50);
```

What happens? What did you expect to happen?
Drawing Rectangles

To draw a rectangle in processing we use the command:
rect(a,b,c,d) where the parameters mean:

(a,b) is the position of the top left corner of the rectangle,
c is the horizontal width of the rectangle and
d is the height of the rectangle

Try:
size(640,500);
rect(250,200,150,100);

To find out how any of the commands in Processing work we go to the Reference: https://processing.org/reference/ (Each language has it’s own reference)

Activity
Find rect() in the reference and copy the example given to draw a rectangle with rounded edges.
Colours in Coding

Colours in coding are represented by a combination of Red, Green and Blue, our three primary colours.

The dots or pixels on the screen have varying amounts of R, G or B, ranging from 0, to the maximum of 255. (16,777,216 colours)

We specify the colour of something in Processing by giving the amounts of red, green and blue in the colour. (R, G, B) e.g. Red is (255,0,0), Green is (0,255,0) and Blue is (0,0,255)

Try changing the colour of the background as in the following:

```
size(250,250);
background(255,0,0);
```
We can use the Colour Selector in Processing to work out the amounts of R, G and B that make up a colour.

Go to Tools -> Color Selector

Experiment with setting the background to different colours!
Other Colour functions

stroke(#) set the colour of the stroke of a line, or the line around a shape
noStroke() disables the stroke (outline)

fill(#) sets the inside colour of a shape
noFill() disables the fill

Try:
size(300,300);
stroke(0,255,0);
fill(255,0,0);
rect(30,20,55,55);
noFill();
rect(100,20,55,55);
Drawing Ellipses

To draw an ellipse in Processing we use the command:

\[ \text{ellipse}(a,b,c,d) \]

where the parameters mean:

- \((a,b)\) is the position of the centre of the ellipse,
- \(c\) is the horizontal width of the ellipse and
- \(d\) is the height of the ellipse

Try:

```java
size(640,500);
ellipse(250,200,150,100);
```

How would we draw a circle?

*Hint: Use the reference.*
Activity: Creating an avatar

By combining shapes we can create an avatar. Try the following:

```javascript
size(400,400);
rect(200,200,40,200);
ellipse(200,140,120,120);
ellipse(162,140,32,64);
ellipse(238,140,32,64);
line(180,300,160,320);
line(220,300,240,320);
```
Activity: Design your own avatar

Design your own avatar on grid paper using a combination of lines, rectangles, ellipses and points.

Work out the dimensions of your shapes that make up your avatar that you will need to code them.

Ask a teacher to check your coordinates before programming your code.

Use `stroke()`, `fill()` and `background()` to add colour to your avatar.
So far we have just drawn static shapes in Processing however this is a bit boring and not how we normally program in Processing.

Processing programs run in two modes: Passive and Active. Programs in Processing usually have two parts: a setup part and a draw part.

The setup method executes once and ‘sets things up’.

The draw() method continually loops and this is how we can code animations in Processing.

A method is a chunk of code.
Three Building Blocks of Programming and algorithms

- Sequence
- Repetition (loops)
- Selection (Conditions/IF-ELSE)

We can implement all the computational algorithms in the entire universe using these three structures
Loops in Processing

From now on all our programs will have a `setup()` function and a `draw()` function:

The syntax (or way we write it) in Processing is:

```java
void setup() {
    // setup code here
}

void draw() {
    // code to loop through here
}
```
Conditional if Statements

Test this out!

https://youtu.be/fVUL-vzrIcM
Variables

We use variables in coding to ‘remember’ values for later.

Variables always have a type and a name (they can’t have spaces in them)

Types include int (integer), float (decimal) and boolean (true or false)

We need to ‘declare’ the variable’s datatype, usually at the beginning of the program

```java
int ball_x_location;
```

In variable names, the computer is very picky, and you must match the ‘case’ of the name (i.e. capital or lower case letters)
int ball_x_location = 0;

void setup() {
  size(400, 400);
  fill(255, 0, 0);
}

void draw() {
  background(0, 255, 0);
  ellipse(ball_x_location, 50, 50, 50);

  ball_x_location = ball_x_location + 1;
}
How does this work?

Remember the `draw()` code is continually running.

Each time the `draw()` method is called, the circle is drawn at a different x location so it looks like it’s moving.

The redrawing of the background goes over the top of the previous ball. This is how we animate things in Processing.

Activity

*How could we change the code to make the ball move faster?*

*How could we make the ball move from the top left of the screen to the bottom right?*
Making the ball move faster

```cpp
int ball_x_location=0;

void setup() {
    size(400,400);
    fill(255,0,0);
}

void draw() {
    background(0,255,0);
    ellipse(ball_x_location,50,50,50);
    ball_x_location = ball_x_location + 5;
}
```
Making the ball move diagonally

```java
int ball_x_location=0;
int ball_y_location=0;

void setup() {
    size(400,400);
    fill(255,0,0);
}

void draw() {
    background(0,255,0);
    ellipse(ball_x_location,ball_y_location,50,50);

    ball_x_location = ball_x_location + 1;
    ball_y_location = ball_y_location + 1;
}
```
Randomness

It’s always good to have randomness in your coding, especially if you are coding a game.

We can now change our starting point to be a random location using the random() function which gives a decimal number within a certain range.

e.g. `random( 0 , width)` gives a random decimal number between 0 and the width of the screen. (*width is a built in variable*)

We need to convert this to an `int` for our ball location so it becomes:

```cpp
int ball_x_location = int ( random (0,width) );
```
Random Starting Point

```cpp
int ball_x_location = int(random(0, width));
int ball_y_location = int(random(0, height));

void setup() {
    size(400, 400);
    fill(255, 0, 0);
}

void draw() {
    background(0, 255, 0);
    ellipse(ball_x_location, ball_y_location, 50, 50);

    ball_x_location = ball_x_location + 1;
    ball_y_location = ball_y_location + 1;
}

Test this out!
```
Random Speed

```cpp
int ball_x_location = int(random(0, width));
int ball_y_location = int(random(0, height));
int speed_x = int(random(0, 5));
int speed_y = int(random(0, 5));

void setup() {
  size(400, 400);
  fill(255, 0, 0);
}

void draw() {
  background(0, 255, 0);
  ellipse(ball_x_location, ball_y_location, 50, 50);
  ball_x_location = ball_x_location + speed_x;
  ball_y_location = ball_y_location + speed_y;
}
```

Test this out!
Bouncing off the edges

Let’s use an ‘if’ statement to make the ball bounce off the edges. The logic is:

\[
\text{if (the ball has hit the edge)} \\
\quad \text{then bounce off.}
\]

We can work out if it has hit the edge by comparing the x-coordinate of the outside edge of the ball to the window width. Then we can reverse it’s direction By multiplying the current speed by -1.

```cpp
// Bounce off the edges
if ((ball_x_location - 50 / 2 <= 0) || (ball_x_location + 50 / 2 >= width))
    speed_x = speed_x * -1;
if ((ball_y_location - 50 / 2 <= 0) || (ball_y_location + 50 / 2 >= height))
    speed_y = speed_y * -1;
```

Test this out!
Reacting to mouse press

There is a special function called `mousePressed()` that gets called when the mouse is pressed. We can use this to stop our ball bouncing when we press the mouse. We can trace the mouse on the screen by two values: `mouseX`, `mouseY`.

```java
void mousePressed() {
    // If the ball is stopped, restore its speed
    if (speed_x == 0 && speed_y == 0) {
        speed_x = saved_speed_x;
        speed_y = saved_speed_y;
    } else {
        // Otherwise, it's moving, so stop it
        // remembering its former speed
        saved_speed_x = speed_x;
        saved_speed_y = speed_y;
        speed_x = 0;
        speed_y = 0;
    }
}
```
Extensions: keyPressed

See if you can change the speed to go faster when you press the up key and slower when you press the down key.

Check out the keyPressed variable in the reference.
Extensions

Exercise:
• Make an ellipse (Circle) that follows our mouse (Use mouseX, mouseY variables).
• Change the circle colour and reduce its size when we click on the page (Use mousePressed in an If statement).
• Use mouseButton variable with its default values (LEFT, RIGHT, CENTER) in an IF statement to detect different buttons of the mouse and change the colour accordingly.