# Lab 1.1 Introduction to Java Programming Using Processing

## Revision:

During your pre-session activity you should have become familiar with the resources at **Processing.org** website (ensure this is bookmarked) and creating and saving various processing sketch files.

Read the webpage on Problem Solving

Support : See top of page for contact details : support tutor, unit tutors

Ask for support while undertaking the following core exercises.

## Learning Objectives

* Familiarity with Processing Environment
  + Writing and saving a program
  + Running a program
  + Correcting syntax errors
* Simple problem decomposition – break problem into smaller chunks
* Program flow (order of commands)
* Using Variables within expressions

## Resources:

open these and have them available for reference

* Lecture Notes (moodle)
* Processing.org website (add to favourites)

**Exercise 1 : familiarisation**

Open ***Processing*** P on your machine. You’ll see a text editor with “sketch” at the top and a text editor below (white background)

//A simple drawing Program

float x;

x = 50;

//screen size 500x500 pixels

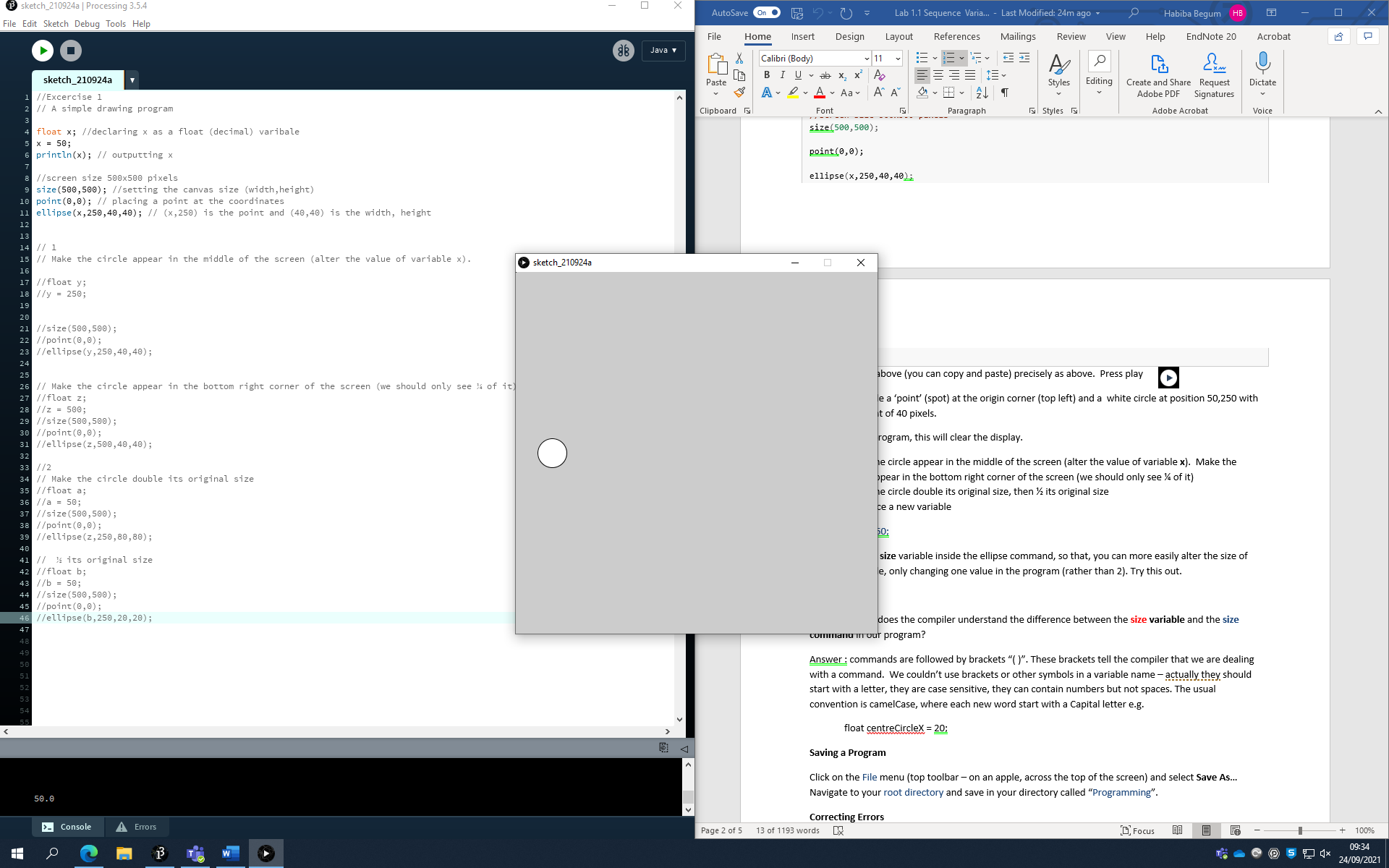
size(500,500);

point(0,0);

ellipse(x,250,40,40);

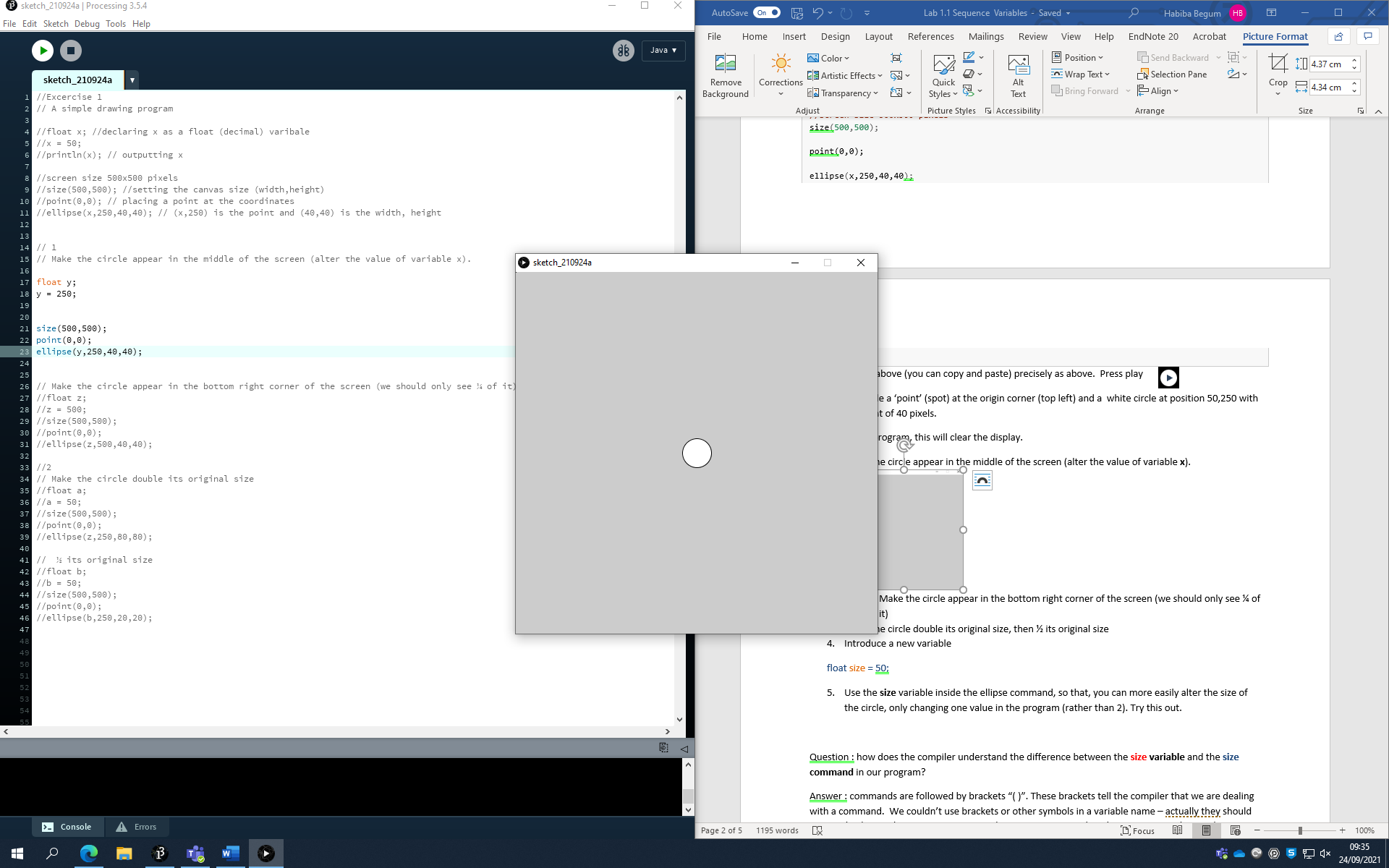
Enter the code above (you can copy and paste) precisely as above. Press play

You should made a ‘point’ (spot) at the origin corner (top left) and a white circle at position 50,250 with width and height of 40 pixels.

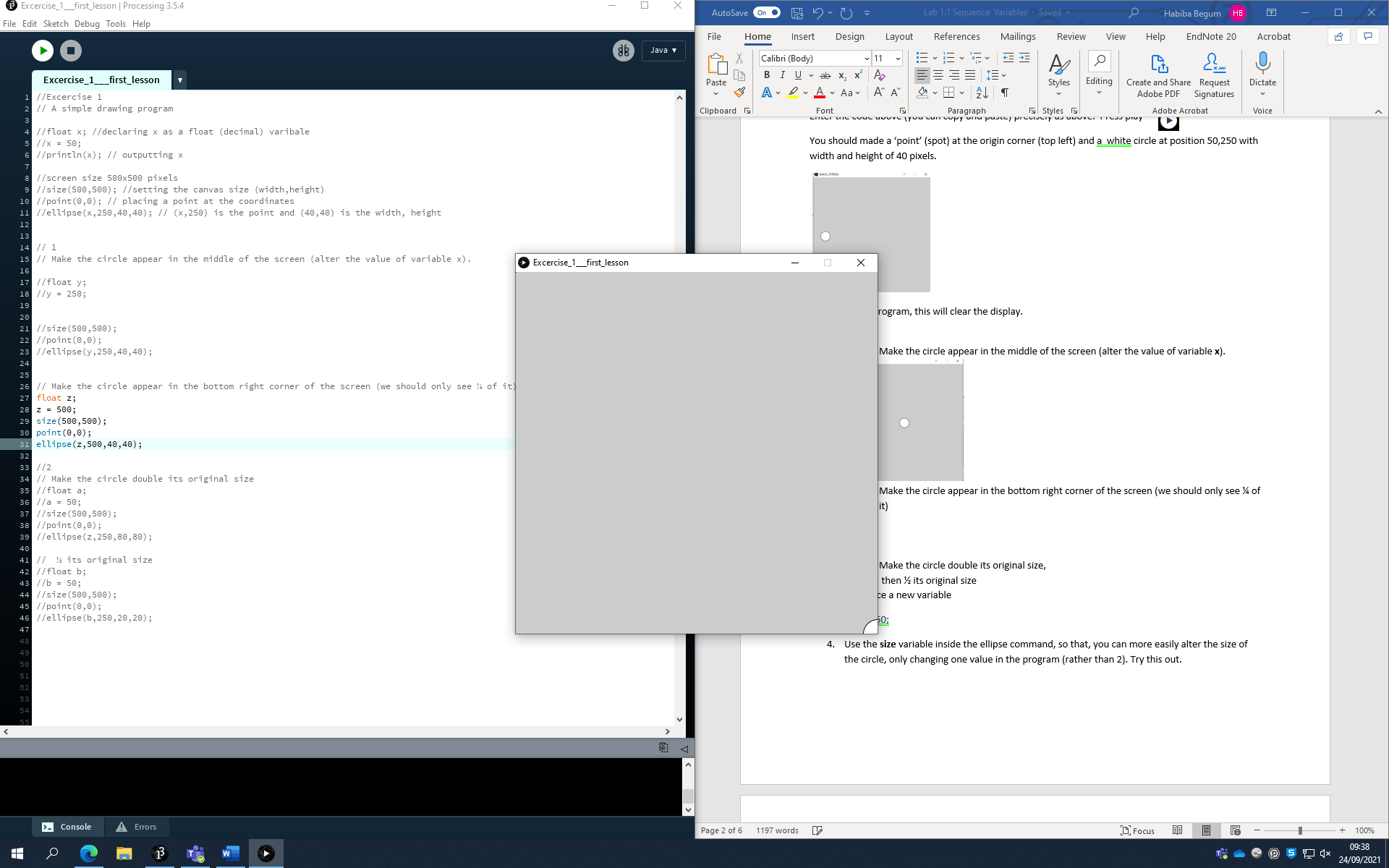


Now stop the program, this will clear the display.

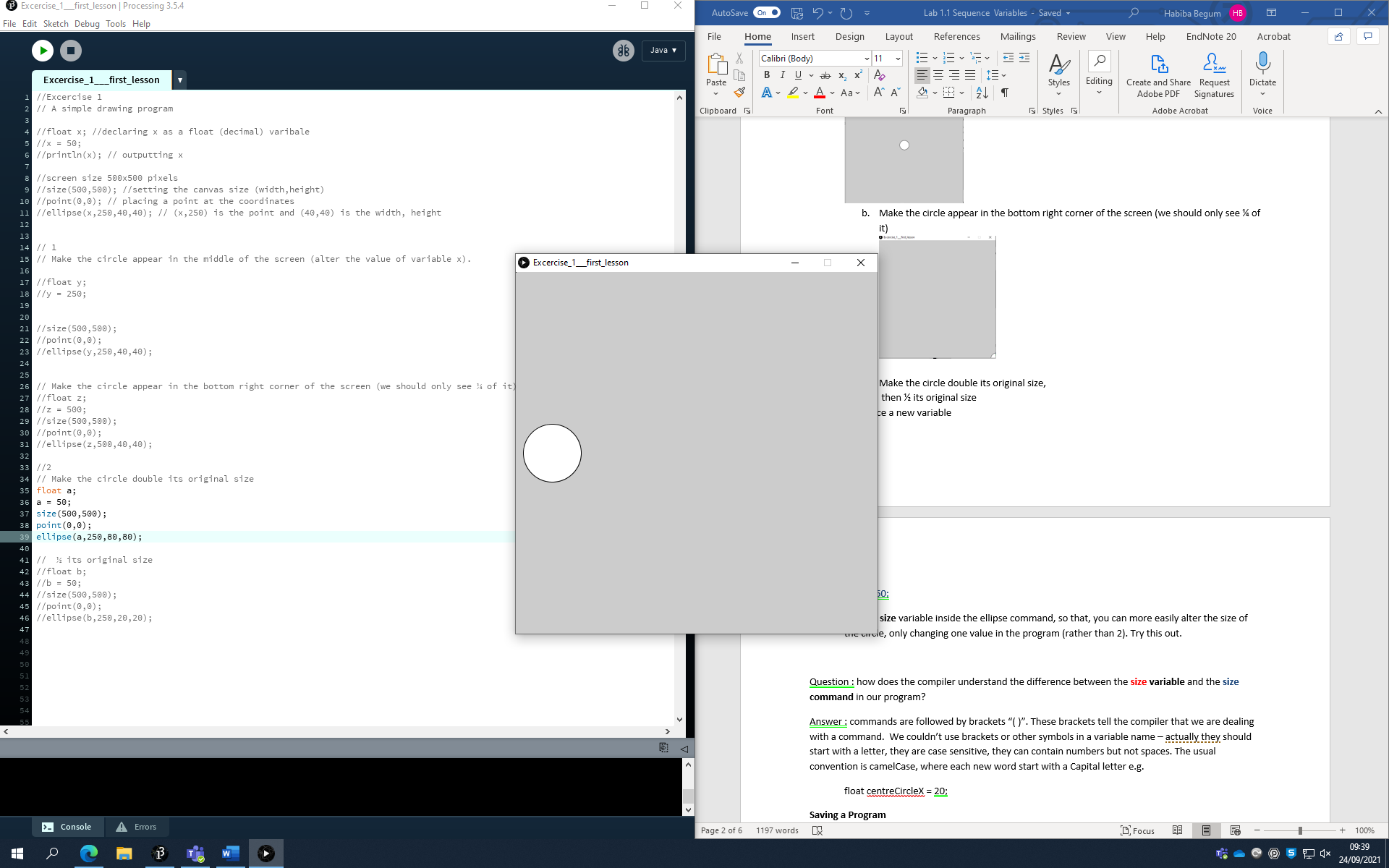
* 1. Make the circle appear in the middle of the screen (alter the value of variable **x**).

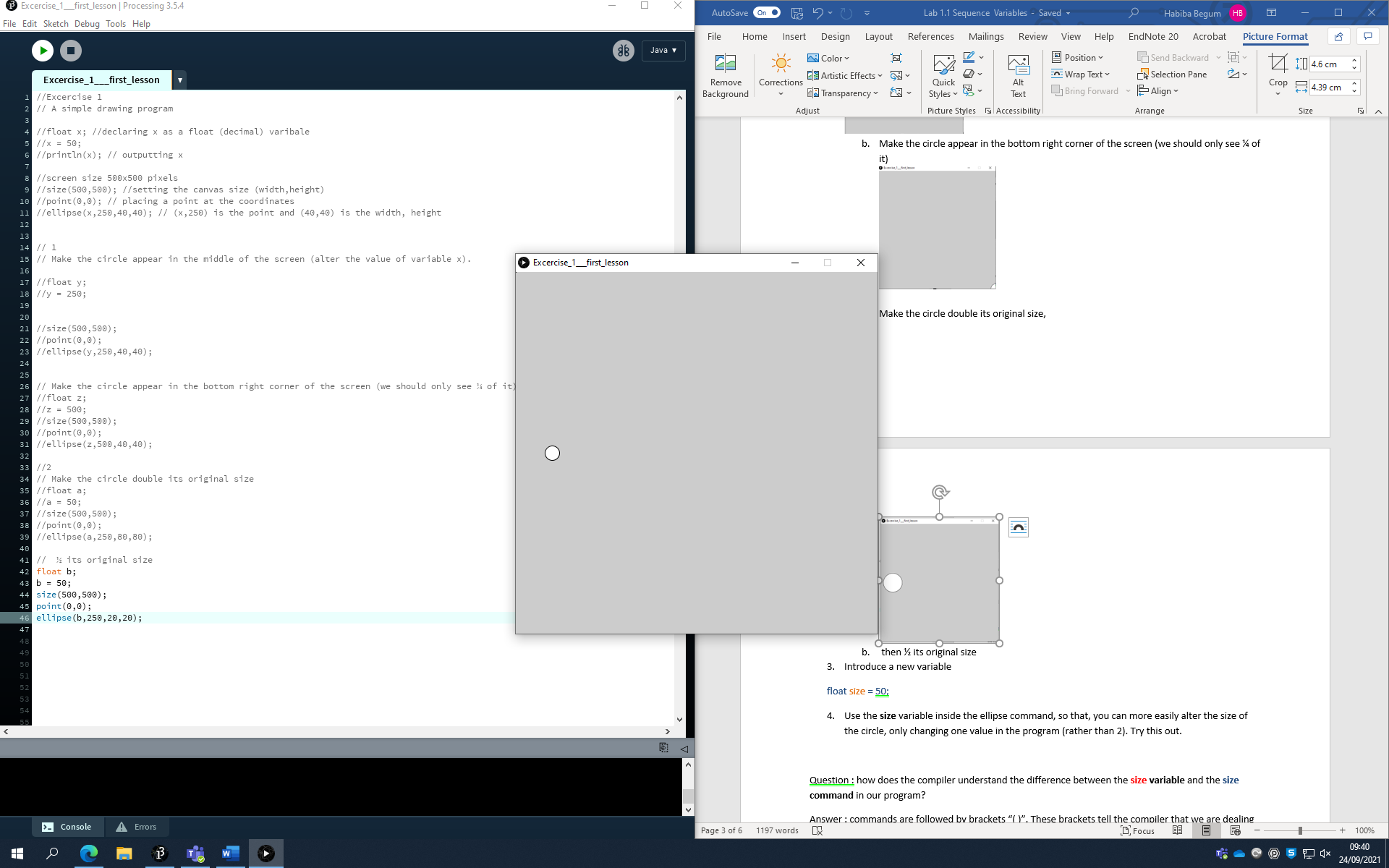


* 1. Make the circle appear in the bottom right corner of the screen (we should only see ¼ of it)



* 1. Make the circle double its original size,



* 1. then ½ its original size
  2. 

1. Introduce a new variable

float size = 50;

1. Use the **size** variable inside the ellipse command, so that, you can more easily alter the size of the circle, only changing one value in the program (rather than 2). Try this out.

Question : how does the compiler understand the difference between the **size** **variable** and the **size** **command** in our program?

Answer : commands are followed by brackets “( )”. These brackets tell the compiler that we are dealing with a command. We couldn’t use brackets or other symbols in a variable name – actually they should start with a letter, they are case sensitive, they can contain numbers but not spaces. The usual convention is camelCase, where each new word start with a Capital letter e.g.

float centreCircleX = 20;

**Saving a Program**

Click on the File menu (top toolbar – on an apple, across the top of the screen) and select **Save As**… Navigate to your root directory and save in your directory called “Programming”.

**Correcting Errors**

When entering or writing your code it’s easy to introduce errors.

Syntax errors are those where a line (or more) of code are not formatted correctly. For example (see image above) I’ve altered our first program a little to introduce an error and pressed play. Processing has told me there is an error either on or before the highlighted line. Try this out, remove a semi-colon (end of command marker) from a line and hit play.

Another common error is to miss one of the parameters to a command.

Try swapping the line size(500,500); for size(500);

Press play and look at the error. Correct this and continue.

we’ll meet other errors as we go.

**Exercise 2 drawing a square**

Draw a square of edge 50 pixels on the screen using only the **line** command, go clockwise. Line takes 4 parameters (2 pairs) as follows

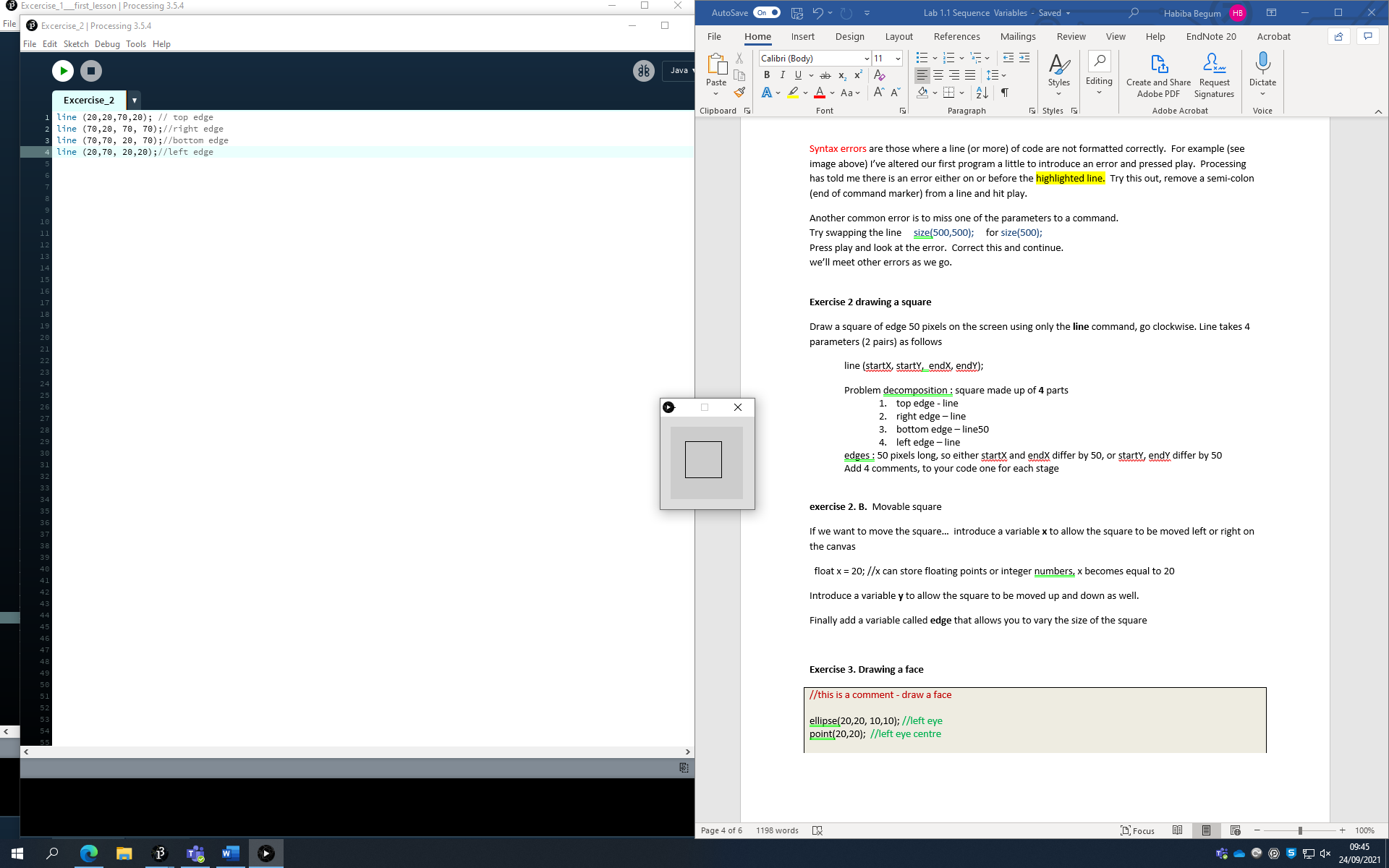
line (startX, startY, endX, endY);

Problem decomposition : square made up of **4** parts

1. top edge - line
2. right edge – line
3. bottom edge – line50
4. left edge – line

edges : 50 pixels long, so either startX and endX differ by 50, or startY, endY differ by 50

Add 4 comments, to your code one for each stage



**exercise 2. B.** Movable square

If we want to move the square… introduce a variable **x** to allow the square to be moved left or right on the canvas

float x = 20; //x can store floating points or integer numbers, x becomes equal to 20

Introduce a variable **y** to allow the square to be moved up and down as well.

Finally add a variable called **edge** that allows you to vary the size of the square

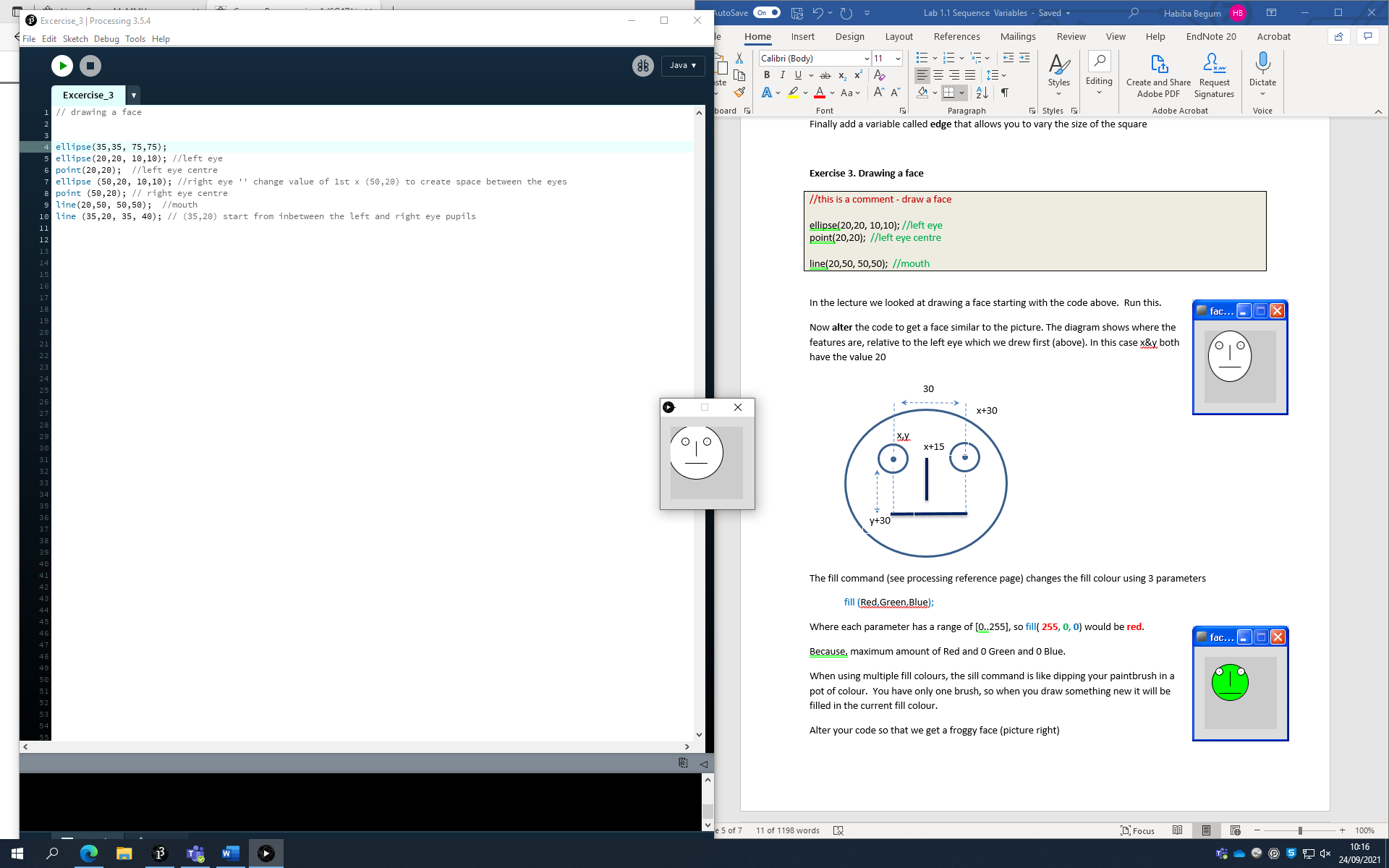
**Exercise 3. Drawing a face**

//this is a comment - draw a face

ellipse(20,20, 10,10); //left eye

point(20,20); //left eye centre

line(20,50, 50,50); //mouth



In the lecture we looked at drawing a face starting with the code above. Run this.

Now **alter** the code to get a face similar to the picture. The diagram shows where the features are, relative to the left eye which we drew first (above). In this case x&y both have the value 20

30

x+30

x+15

x,y

y+30

The fill command (see processing reference page) changes the fill colour using 3 parameters

fill (Red,Green,Blue);

Where each parameter has a range of [0..255], so fill( **255**, **0**, **0**) would be **red**.

Because, maximum amount of Red and 0 Green and 0 Blue.

When using multiple fill colours, the sill command is like dipping your paintbrush in a pot of colour. You have only one brush, so when you draw something new it will be filled in the current fill colour.

Alter your code so that we get a froggy face (picture right)

Experiment with the colours of the face and eyes using the fill command parameters.

Save your program.

**Exercise 4** . We are going to amend our froggy face to allow it to be positioned at an **x** and **y** location. Introduce 2 global variables, **x** and **y** to the top of your code and set them to (100,100). Amend the appropriate commands replacing fixed values with variables. Now it would make more sense if x and y were the centre of the face (not the left eye). So nose would be a line(x, y-?, x, y+?), i.e. from above the centre to below the centre of the face.

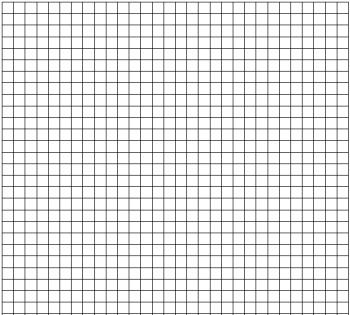
**Exercise 5**– Alter your code so that the face can be scaled as well as positioned, depending on **x**,**y** & **size** variables. You should introduce new variables to ease the coding.

**Exercise 6. Drawing a Stick Person of variable size**

Using the 3 variables **x**,**y**,**size** and only the line and ellipse drawing commands, you must draw a stick person similar to the diagram below (with the same relative sizes and positions of each part) where x,y is the centre of the head and the value of size the length of the torso (centre of head down). Other sizes can be seen by counting the number of graph squares relative to size. You SHOULD add extra variables as required to store suitable starting points for the arms, legs etc.

Your code should be clearly commented to show what each block of commands is drawing.

Experiment with the values of size, x and y to draw different sized people in different places on the screen (remember that processing will allow you to draw off the screen, so if nothing appears alter x and y).



size

size

Save your program.

**Extension exercise**, Add the scalable face code to your stick person