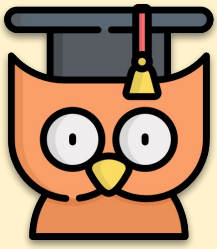
[the academy_of_code]

Grade 4
Unit 3

Welcome to The Academy of Code, and for returning students, welcome back!

In this block of lessons we will explore slightly more advanced topics than those we met in the previous unit, including nested if statements, booleans and loading images. These concepts will build on what you have learned in the previous unit so it is vitally important that you have fully completed Unit 2 and have more or less understood it all!

Happy coding!
Diarmuid

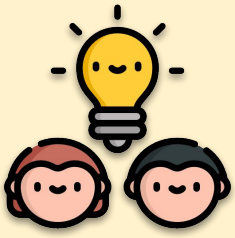


Learning outcomes

What we will learn in this lesson

Expert Tip

Top Advice from the Pro's

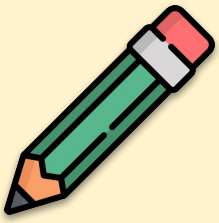


Thinking Time

Think about what is asked

Discussion Time

Discuss/Work with a partner

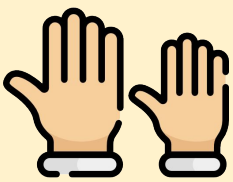


To Do

Lets Get Coding!

Mouse

Click/Double Click



Ask your Tutor

Put up your hand when you need

Design

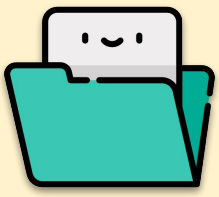
Time to get Creative

Checklist

Check over your work

Challenges

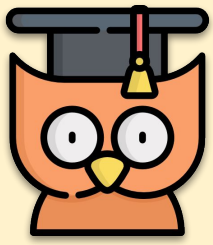
Extra work or Top Marks



Save

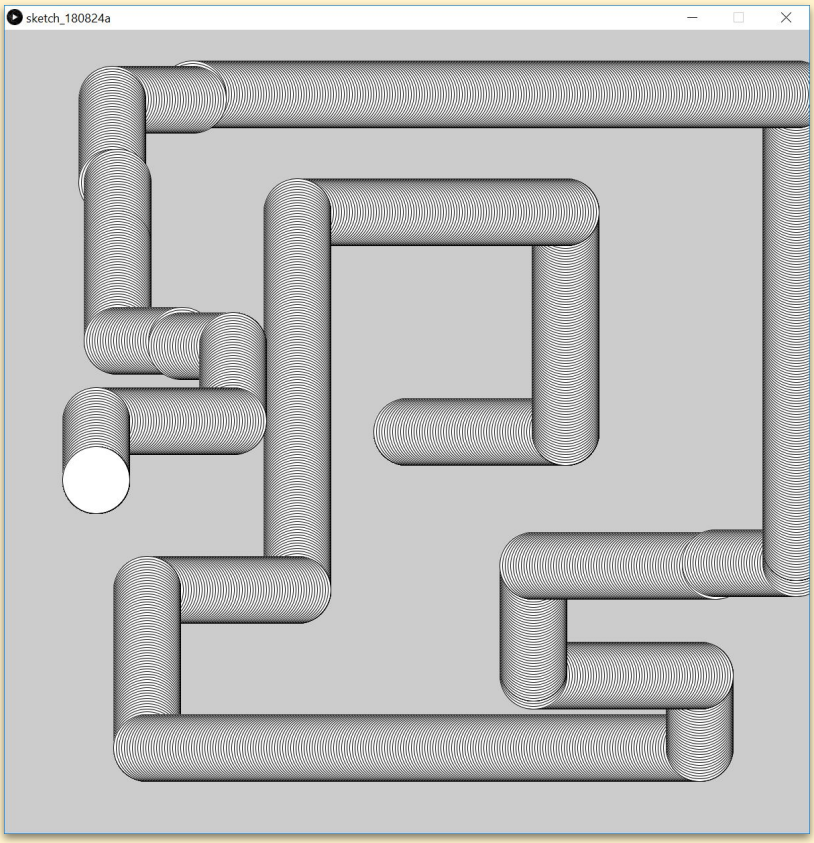
Save your work - MOST IMPORTANTLY

Lesson 1 - IF Statements III



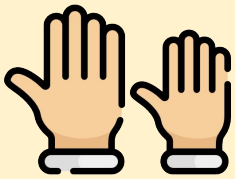
Learning Outcomes:

- Have an understanding of advanced **if** statement concepts, e.g. nested **if** statements.
- Change the location and colour of shapes using the keyboard.
- Use different mouse buttons to run seperate pieces of code.

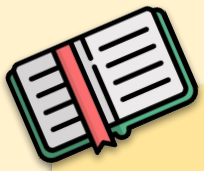


Revision topics before starting make sure you know:

- ✓ Declaring and initializing variables.
- ✓ Using **void setup()** & **void draw()**.
- ✓ Using **if** statements.
- ✓ Using **keyPressed**.



REMEMBER: Put up your hand. We love to help!



What are nested if statements?

Nested if statements are basically if statements **within** if statements. Let's take a simple case:

```
if (keyPressed) { // test all keys on keyboard
  if (key == 'd') { // test the 'd' key
    ballXPos += 1; // increase variable by 1
  }
}
```

The code inside the if statements "ballXPos += 1;" will only execute if the conditions in **both** if statements are true. In this example, if **any** key on the keyboard has been pressed, the second if statement will then be tested. If the 'd' key has been pressed, then the code will execute.

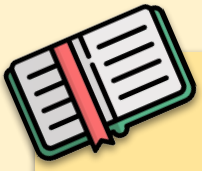
Let's get coding!

1 Create a new Processing sketch and write the following code:

```
float ballXPos = 600;

void setup() {
  size(800, 800);
}

void draw() {
  background(255);

  ellipse(ballXPos, height/2, 100, 100);

  if (keyPressed) {
    if (key == 'd') {
      ballXPos += 4;
    }
  }
}
```

Thinking Time

What do you think this code does?



else if Statements

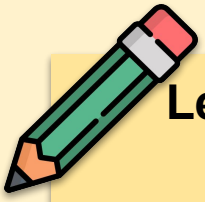
In the previous unit we were introduced to **else** statements that can be used **along with if** statements. We are now going to go one step further and introduce **else if** statements that can be used along with **if** & **else** statements.

else if statements are used when more conditions are needed. They are always used **after** an if statement and their condition will only be tested if the condition in the previous **if**/**else if** statement was **not true**.

Anatomy of an else if statement

```
if (keyPressed) {
  if (key == 'd') {
    fill(255, 255, 0); // yellow
  }
  else if (key == 'a') {
    fill(0, 255, 0); // green
  }
  else if (key == 'w') {
    fill(255, 0, 0); // red
  }
  else {
    fill(255); // white
  }
}
```

If the condition inside the **if** statement is **true**, run the code inside the **if {code block}**
Otherwise, move onto the next **else if** statement and see if the condition is **true**.
Or **else**, if none of the conditions are **true**, run the code inside the **else {code block}**



Let's get coding!

1. **Add** code to make the circle move to the **left** when the '**a**' key is pressed.

2. **Add** code to make the circle move to **up** when the '**w**' key is pressed and **down** when the '**s**' key is pressed.

Be Careful!

The more **if**/**else if**/**else** statements you have, the more brackets you will have. Be careful that you put the correct code inside the correct brackets!

Now would be a great time to save your sketch if you haven't already.

Extra Challenge

Add a few more keys to change the colour of the circle and the colour of the background.

Extra Challenge

Make it so you see only one circle at a time (hide the trail!).

Let's get coding!

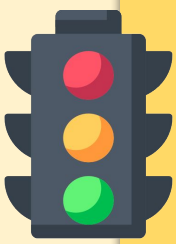
1. **Add** the following code to the **start** of your void **draw**():

```
if (mousePressed) {
  if (mouseButton == LEFT) {
    background(0, 0, 255);
  }
}
```

2 **Add** code to make the background change to a different colour when the **right** mouse button is pressed.

Extra Challenge

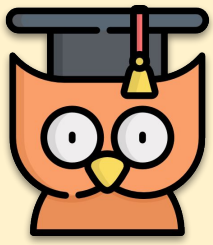
Change your code so that the size of the circle will increase when the right mouse button is pressed and decrease when the left mouse button is pressed.



At this point, you should know how to:

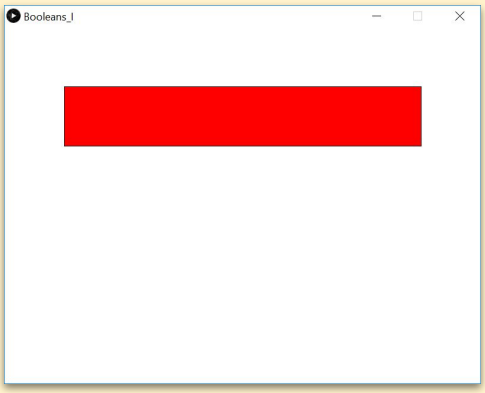
- Nest **if** statements inside one another.
- Use curly brackets **{}** to separate code blocks.
- Use different keys on the keyboard and different buttons on the mouse to execute code blocks.
- Use **else if** statements where appropriate.

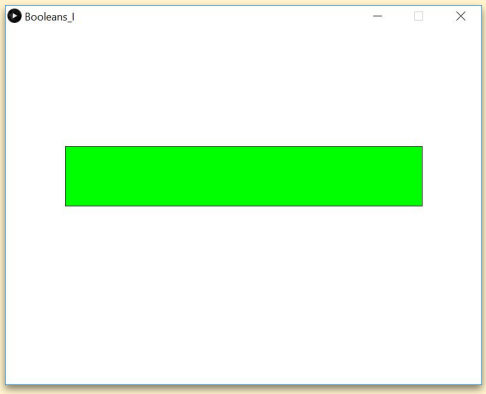
Lesson 2 - Booleans I

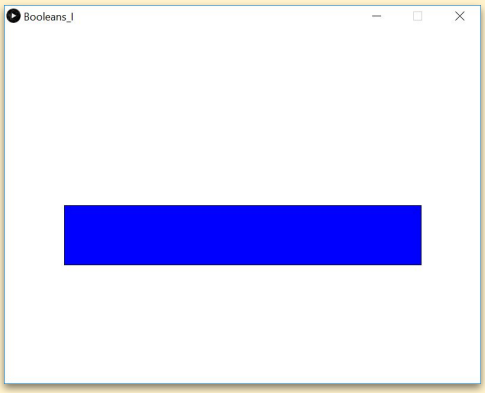


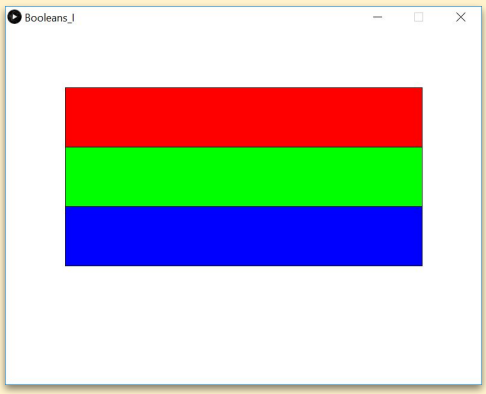
Learning Outcomes:

- Have a basic understanding of **booleans** and how they are used in coding.
- Know how to toggle the value of a **boolean**.



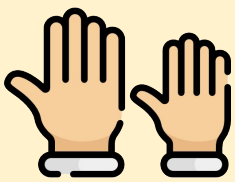






Revision topics before starting make sure you know:

- ✓ Declaring and initializing variables.
- ✓ Using **void setup()** & **void draw()**.
- ✓ Drawing rectangles.
- ✓ How to use **fill()**.
- ✓ Using **if** statements.



REMEMBER: Put up your hand. We love to help!

What are booleans?

A **boolean** is a type of variable. We have also worked with the **int** and **float** variable type.

While **int** and **float** variables contain any *numbers*, a **boolean** can only be **true** or **false**. We will often use them as a type of switch - they will be **true** if they are on, or **false** if they are off.

Like a switch, we can use a **boolean** to turn things on and off. In this example we're going to turn different shapes on and off in our sketch, as in the example above.

How do we use them?

There are 3 simple steps to using **booleans**:

1. We need to **declare** our **boolean** variable, give it a **meaningful name** and **initialize** it by giving it an initial value.
2. Test our **boolean** to see if it is **true**/**false**.
3. Set our **boolean** to **true**/**false**.

Did you know?

This is not actually the first time we have come across **booleans**! We have seen them before in the previous unit when we were using **keyPressed** and **mousePressed**. These are actually **boolean** variables because they can only be **true** or **false**.

Let's get coding!

1. Create a new Processing sketch and at the **top** of your text editor, write this line:

```
boolean redOn = false;
```

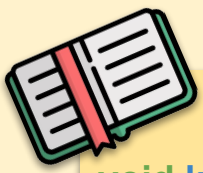
Declares variable "**redOn**" and **initializes** it with a value of **false**

2. Inside **void draw()**, use an **if** statement to test if "**redOn**" is **true**, if it is, then draw a red rectangle.

```
void draw() {
  background(255);
  if (redOn == true) {
    fill(255,0,0);
    rect(100,100,600,100);
  }
}
```

3. At the **bottom** of your text editor outside of **void setup() and void draw()**, make a new function called **void keyReleased()**, test if the 'r' key has been pressed, if it has then set our **boolean** to true.

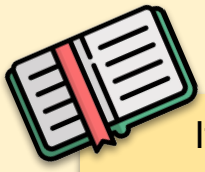
```
void keyReleased() {
  if (key == 'r') {
    redOn = true;
  }
}
```



void keyReleased() will execute its **{code block}** when a key has been **released**, so it will only run **once**. Using **keyPressed** will execute the code multiple times for as long as the key is being held down.

Expert Tip

Instead of writing `if (redOn == true)` this can be **shortened** to `if (redOn)`



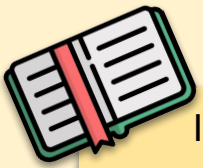
If you've done the steps correctly, you should see a red rectangle on the screen when the 'r' key is pressed.

Now would be a great time to save your sketch if you haven't already.

Let's get coding!

1. Add code for green and blue rectangles. Pressing 'g' should turn on the green rectangle, 'b' should turn on the blue rectangle.

2. Add code to turn the rectangles on and off, when the keys are pressed.
   Hint: To do that we need to set the **boolean** (eg. "redOn") to its opposite value. If it's true make it false, if it's false make it true. Putting '!' in front of a boolean makes it the opposite.



If you've done the steps correctly, you should be able to toggle each of the three rectangles on and off using the keys.

Now would be a great time to save your sketch if you haven't already.

Congratulations

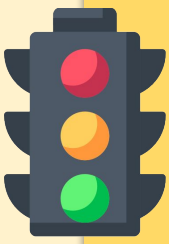
You have made your first toggle switch using booleans!

Extra Challenge

Change the position and colour of the boxes to make the Irish flag.

`Be Careful!`

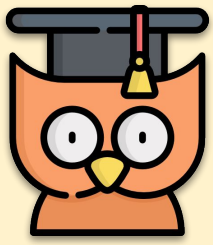
Don’t forget to change your variable names to suit the changes you have made to your code. This will avoid confusion!



At this point, you should know how to:

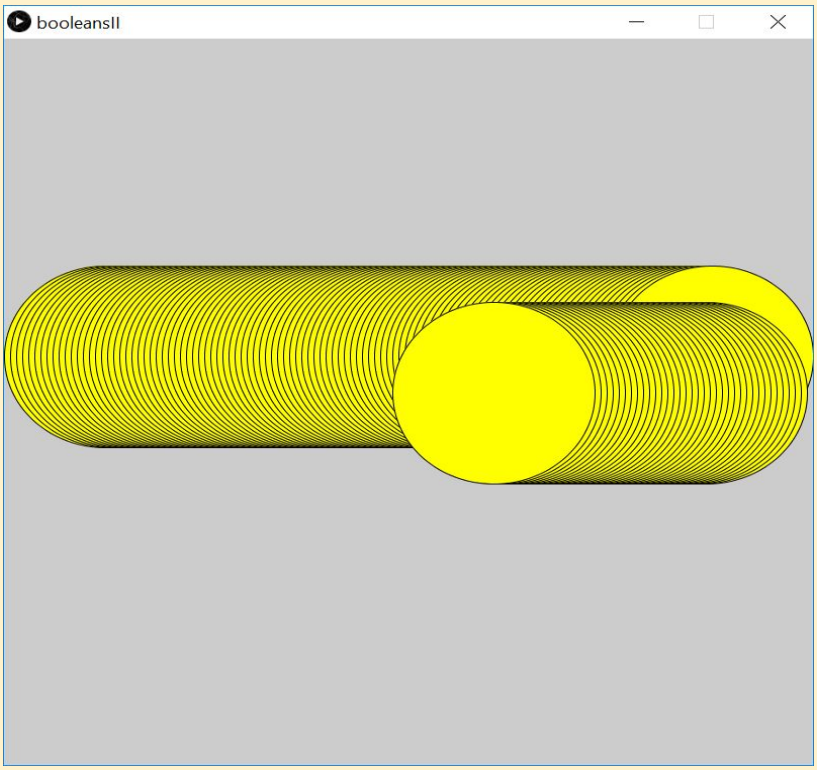
- Declare and initialize a **boolean** variable.
- Test if the variable is **true** or **false**.
- Turn the **boolean** “on” and “off”.

Lesson 3 - Booleans II



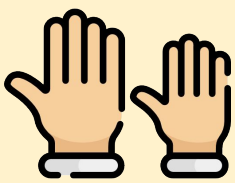
Learning Outcomes:

- Reinforce our understanding of **booleans** and how they are used.
- Get familiar with different scenarios where **booleans** would be appropriate to use.

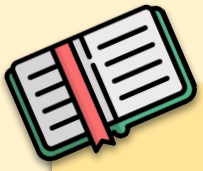


Revision topics before starting make sure you know:

- ✓ Using **booleans**.
- ✓ Using **if** statements.
- ✓ Moving shapes.
- ✓ Setting end conditions.



REMEMBER: Put up your hand. We love to help!



More About booleans

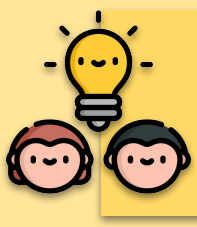
As we have seen in the previous lesson, **booleans** can have two values, **true** and **false**. These values are obviously opposites to one another.

In our code we can use **booleans** for things that are the opposite of one another. We did this in the previous lesson with the coloured rectangles, when the **boolean** "**redOn**" is **true**, show the rectangle, or else if it is **false**, do not show the rectangle. As you can see, these two things are opposites of one another.

Let's get coding!

1 Create a new Processing sketch and copy the following code:

```
boolean moveBallLeft = false;
float ballHozPos = 100;

void setup(){
  size(800,800);
}
void draw(){
  fill(255,255,0);  // colour yellow
  ellipse(ballHozPos, height/2, 200, 200);  // make ball

  if(ballHozPos >= 700){   // if ball hits right side
    moveBallLeft =  true;
  }
  if(moveBallLeft == true){
    ballHozPos -= 4;   // move left
  }
  else {
    ballHozPos += 4;   // move right
  }
}
```



Thinking Time

What do you think this code does?

2 Make it so you only see one ball at a time.

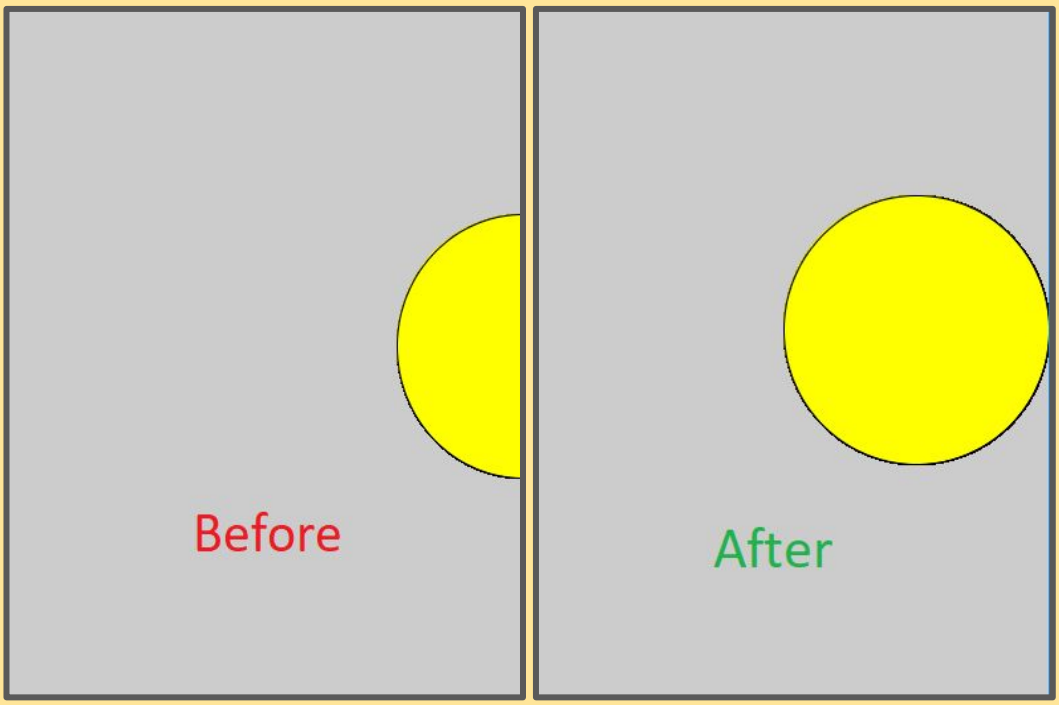
3 Make the ball bounce off the **left side** of the screen and move to the right.

Hint: You should only need to add one **if** statement for this.

Be Careful!

We currently have a **boolean** "**moveBallLeft**" that will make the ball move to the left when it is **true**. If we want to make the ball move to the right if it has hit the left side of the screen, a mistake some people make is to make another **boolean** called "**moveBallRight**". However, from what we have learned up to now, this clearly isn't the best solution.

4 Change the code so the edge of the ball bounces off the two sides, not the centre of the ball.



Now would be a great time to save your sketch if you haven't already.

Congratulations

You have displayed your first image that moves with user input!

Extra Challenge

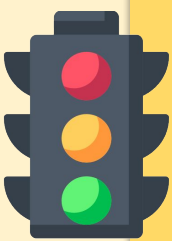
Change the code so that a **rectangle** bounces **up** & **down** the screen.

Hint: Change the variable names where appropriate so you don't get confused.

Extra Challenge

Make the rectangle slow down each time it hits the bottom of the screen. Ensure that the speed does not go below 0!

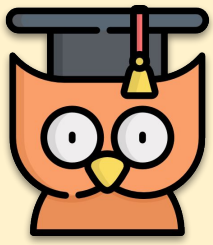
Hint: You will need to use a variable for the speed of the rectangle.



At this point, you should know how to:

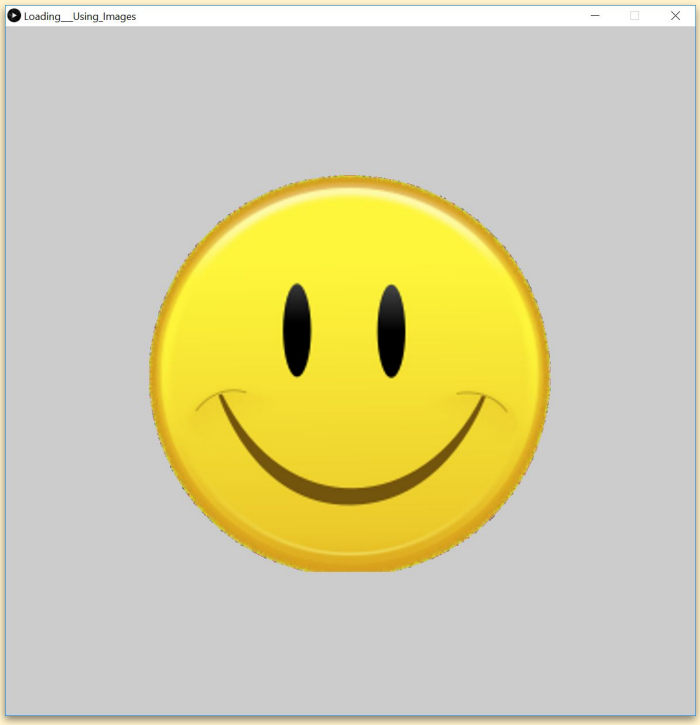
- Decide whether **boolean** variables are appropriate to use in different cases.
- Use **boolean** variables.
- Appropriately name **boolean** variables.

Lesson 4 - Loading & Using Images



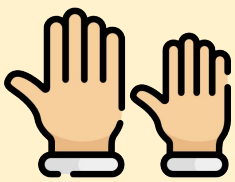
Learning Outcomes:

- Know how to save an image onto your USB stick.
- Know how to load an image into a **PImage** variable.
- Know how to resize an image.
- Know how to display the image in a Processing canvas.
- Know how to move an image around using user input



Revision topics before starting make sure you know:

- ✓ Naming and saving files.
- ✓ Declaring and initializing variables.
- ✓ User input using the mouse & keyboard.

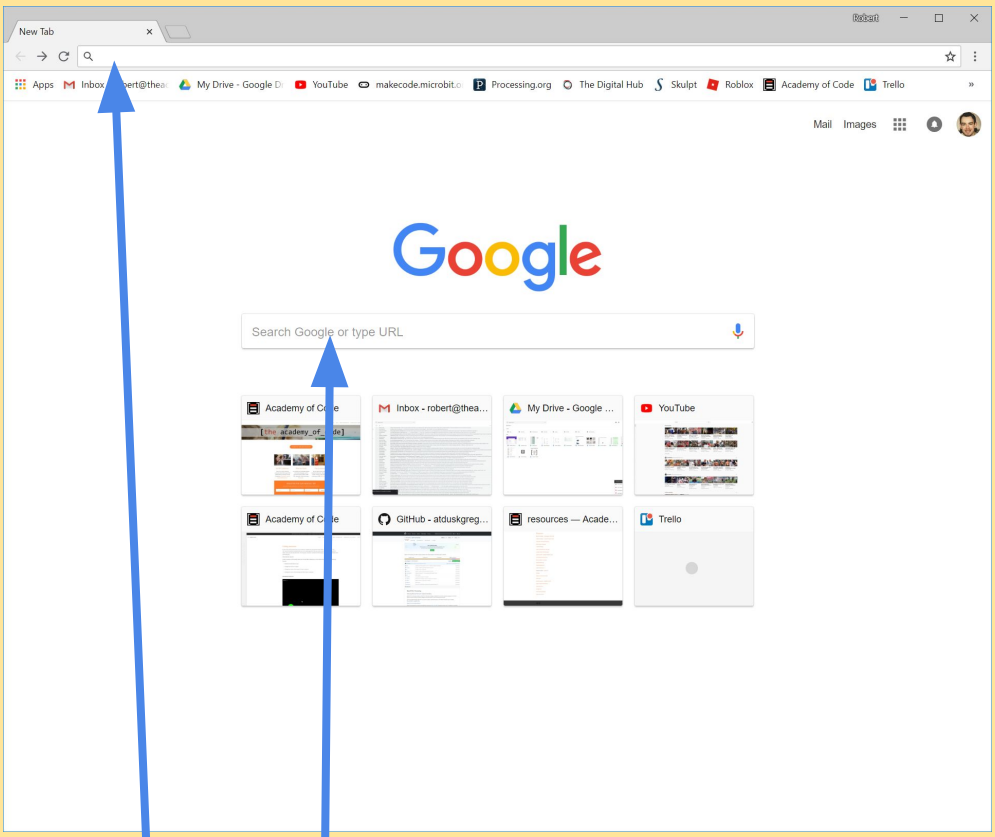


REMEMBER: Put up your hand. We love to help!

Saving an Image

If you are looking for an image to use in a program, whether it be for a background or a sprite etc. you can usually find what you are looking for on **Google**. Here are the steps to save an image you find on Google:

1. Open a new tab in Google

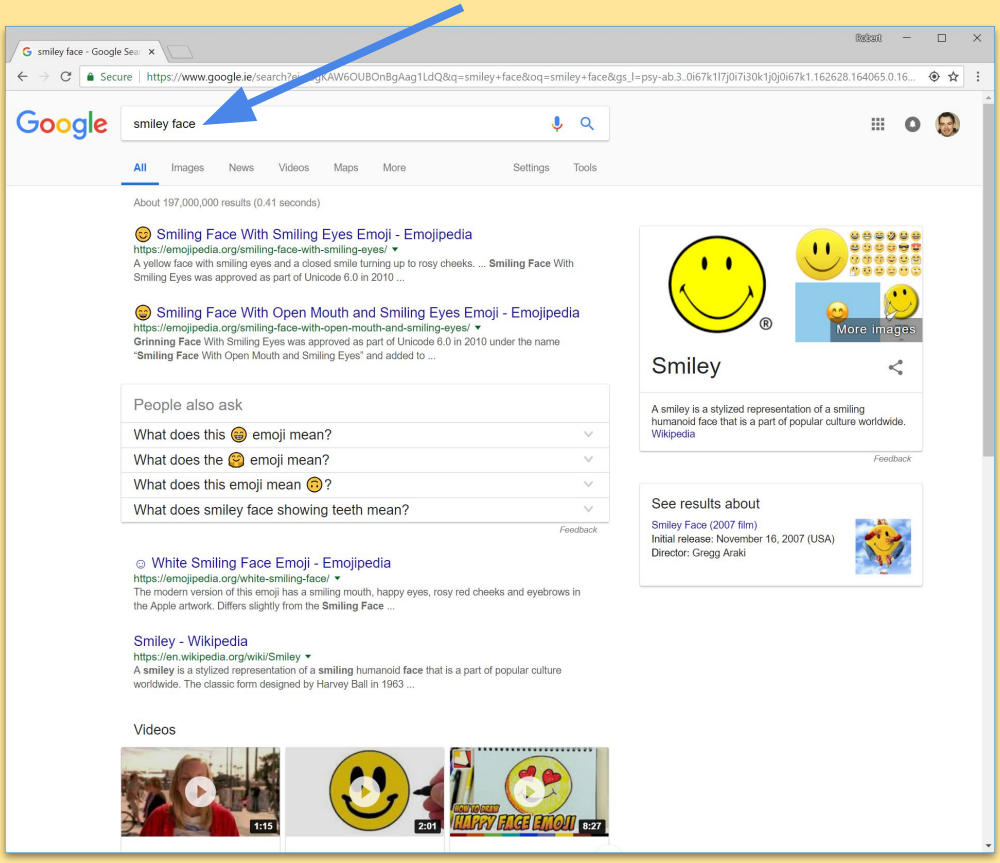


2. Type the keyword(s) into the Google search bar or address bar

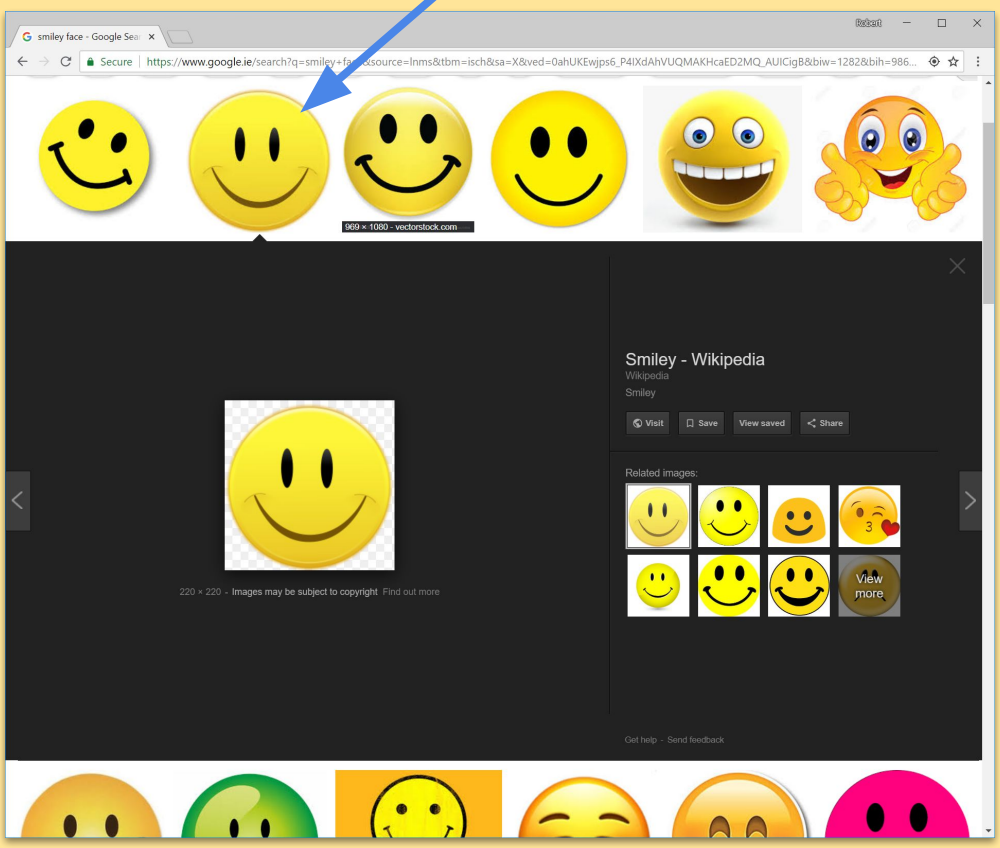
Expert Tip

When you are searching for an image on Google you don't have to be very specific. If you are looking for an image of a smiley face, you just need to type "smiley face" into the search bar. It's usually a waste of time to type something like "a picture of a smiley face".

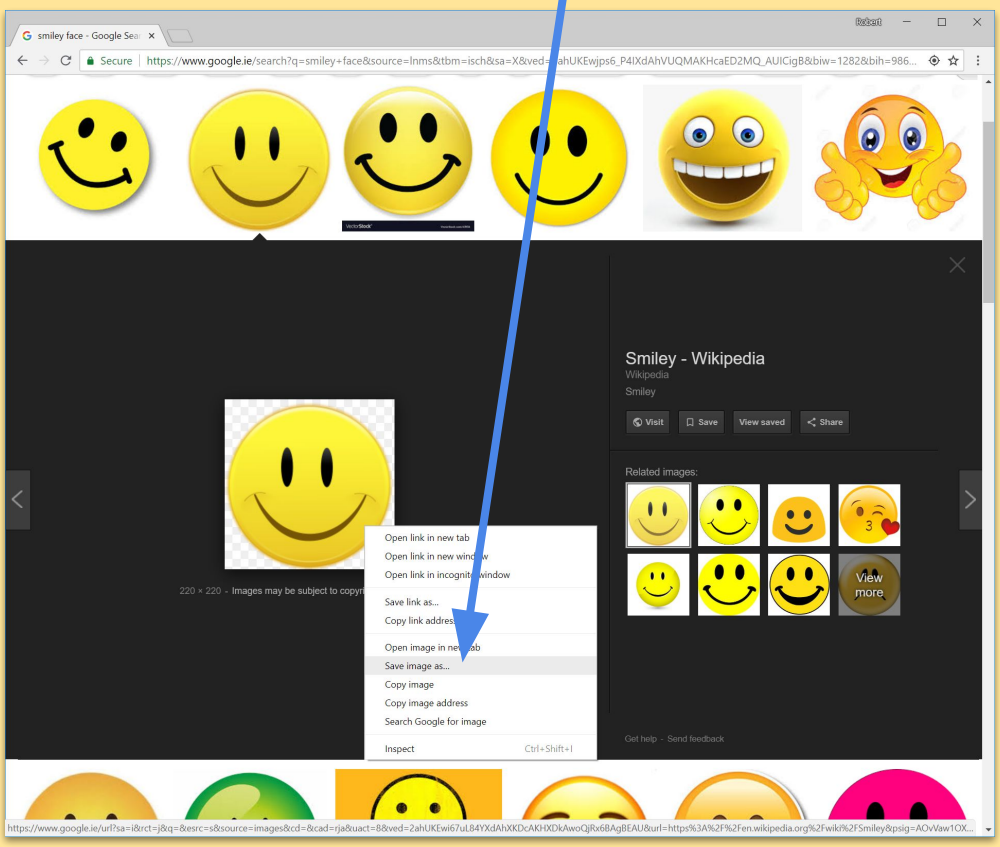
3 Click into the “Images” section and Google will find all of the images on the internet relating to those keywords.



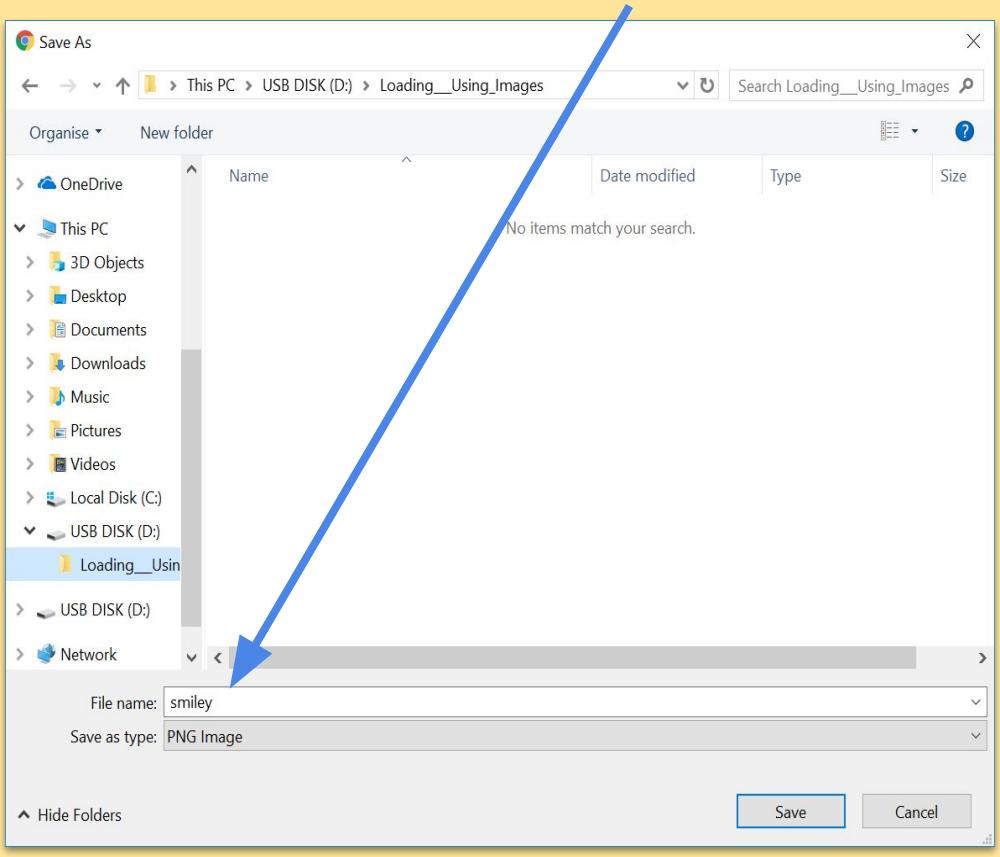
4 Click on the image that you want to use

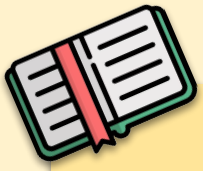


5 Right-click on the image and click “**Save image as…**”



6 Locate the lesson folder on your **USB** stick, double-click into it, give the image a **new file name** and click **save**.





Types of Images

There are several different image file types you can use in Processing, the main ones being: JPEG (**.jpg**), PNG (**.png**), GIF (**.gif**)

If you are using an image as a sprite in a game you are making for example, the best type of image to use would be a **PNG** image because they can have a **transparent background** (no white box around the image).

When searching for an image to use on Google, you will know if an image has a transparent background if there are checkered boxes in the background, like in the example below.

Be Careful!

Just because you can't see the background of a PNG image doesn't mean that it's not there. PNG images with transparent backgrounds are still rectangular or square.

How do we use images in our Program?

There are 3 simple steps to using images:

1. We need to **declare** our **PImage** variable and give it a **meaningful name**, e.g. “backgroundImage”, “playerSprite”, “enemySprite”, etc.
2. Load the image into our **PImage** variable in **void setup()**.
3. Resize the image if necessary.
4. Display the image in **void draw()**.

Let’s get coding!

1. Create a new Processing sketch and save it as “**Lesson4_Images**”.
2. Find an image you want to use on Google and save it into your lesson folder. **This should not take longer than 2-3 minutes!**
3. Declare your **PImage** variable at the **top** of your text editor.

```
PImage myImage;
```

4. Inside **void setup()**, load the image into your variable and resize it if it is too big or too small.

```
void setup() {
  size(800,800);
  myImage = loadImage("myImageFileName.png");
  myImage.resize(150,150);
}
```

The part in the quotation marks (“”) is the image **file path**. This will be whatever you have saved your image as.

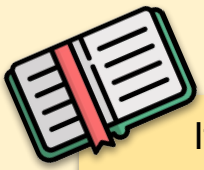
5 Inside **void draw()**, display the image using the **image()** function.

```
void draw() {
  image(myImage,400,400);
}
```

The two numbers in the **image()** function after the **PImage** variable are the **x-coordinate** and **y-coordinate** of the image.

Expert Tip

Just like **rect()**, images start at the **top-left corner**. However, we can change this so the origin is at the **centre** (just like an ellipse) using imageMode(CENTER); in **void setup()**, just after you have loaded the image.



If you've done the steps correctly, you should see the image displayed on the canvas.

Now would be a great time to save your sketch if you haven't already.

Let's get coding!

1. Make the image move around, following the **mouse**.
2. Change the code to make the image move using the '**w**', '**a**', '**s**', '**d**' keys on the **keyboard**.

Now would be a great time to save your sketch if you haven't already.

Congratulations

You have displayed your first image that moves with user input!

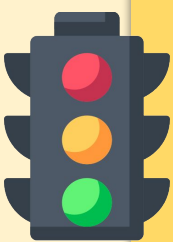
Extra Challenge

Display a **second** image on the screen.

Extra Challenge

Make the second image move around using the **keyboard**. However it must be always underneath the first image.

Hint: Use the same variables as the first image and change the position with offsets, e.g. yPos + 100.



At this point, you should know how to:

- Save an image from Google onto your USB stick.
- Declare a **Pimage** variable.
- Load and resize and image.
- Display the image on the canvas.
- Move the image using the keyboard and mouse.