Programming Fundamentals 1

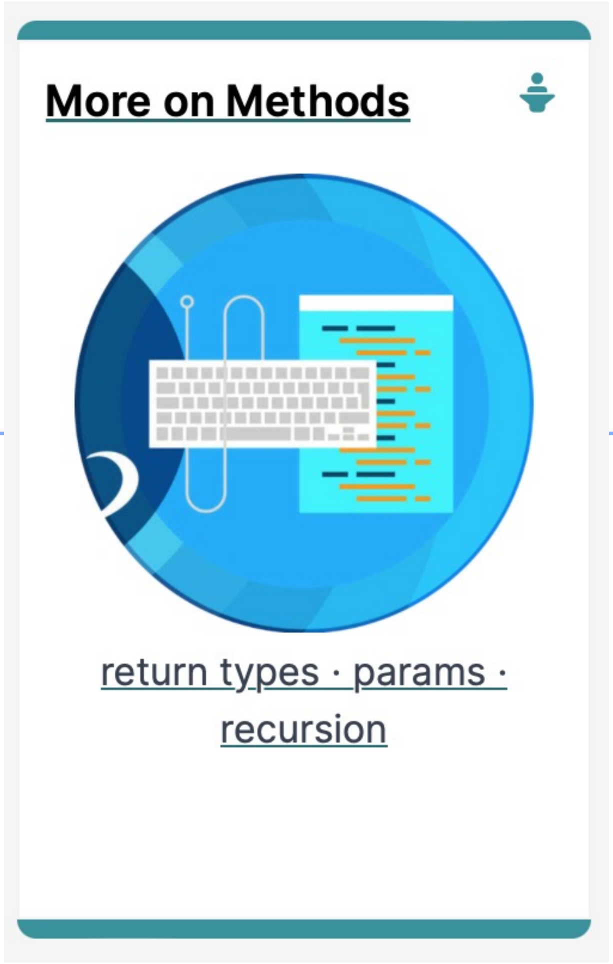
Produced by Mr. Dave Drohan (david.drohan@setu.ie)
Dr. Siobhán Drohan
Ms. Mairead Meagher

Department of Computing & Mathematics
South East Technological University
Waterford, Ireland

setu.ie

Introduction to Processing

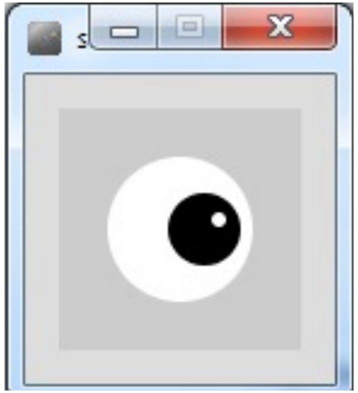
More on writing your own Methods

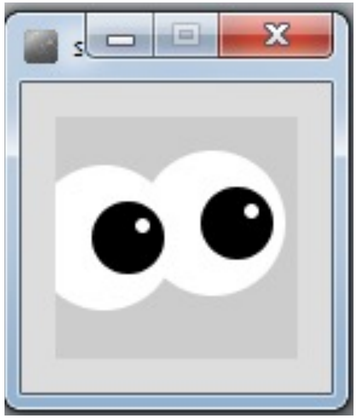


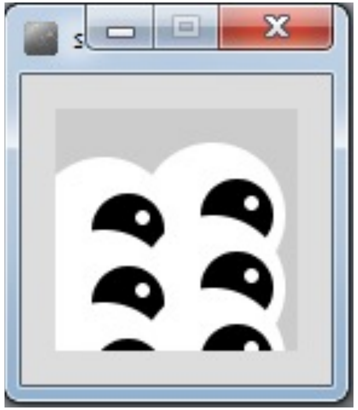
Agenda

- ❑ Method example: Eyes
- ❑ Method example: X’s
- ❑ Overloading methods.
- ❑ Method example: Celcius / Farenheit Converter.
- ❑ Recursion.

Method example: Eyes



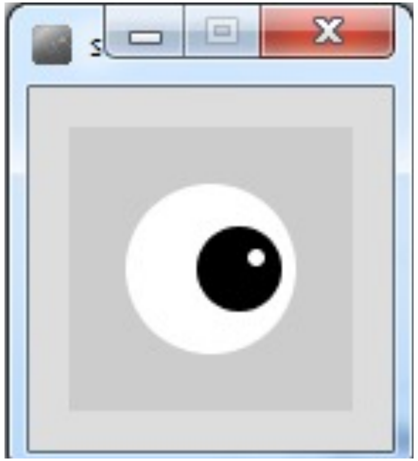




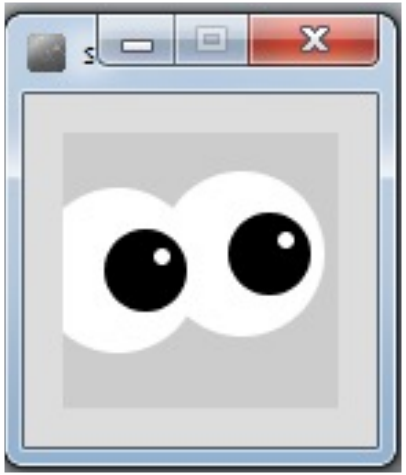
Example 6.1 – Drawing a single eye

```
void setup()
{
  size(100,100);
  noStroke();
}
```

```
void draw()
{
  background(204);
  fill(255);
  ellipse(50,50,60,60);
   //outer white circle
  fill(0);
  ellipse(50+10, 50, 30, 30);
   //black circle
  fill(255);
  ellipse(50+16, 46, 6, 6);
   //small, white circle
}
```



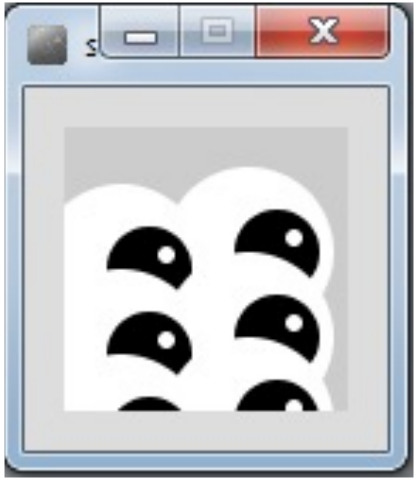
What if we wanted to draw two eyes?



Each eye takes six lines of code to draw.

```
void draw()
{
  background(204);
  //Right eye
  fill(255);
  ellipse(65,44,60,60);       //outer white circle
  fill(0);
  ellipse(65+10, 44, 30, 30);    //black circle
  fill(255);
  ellipse(65+16, 44-5, 6, 6);    //small, white circle
  //Left eye
  fill(255);
  ellipse(20,50,60,60);       //outer white circle
  fill(0);
  ellipse(20+10, 50, 30, 30);    //black circle
  fill(255);
  ellipse(20+16, 50-5, 6, 6);    //small, white circle
}
```

What if we wanted to draw six eyes?



Are we going to repeat the six lines of code SIX times?

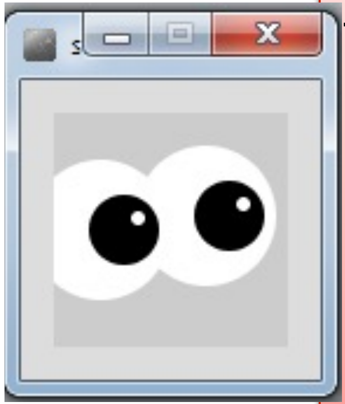
What if we wanted to draw 100 eyes?

→ 600 lines of code!

Example 6.2 – Drawing two eyes

```
void setup()
{
   size(100,100);
   noStroke();
}
```

```
void draw()
{
   background(204);
   eye(65,44);
   eye(20,50);
}
```



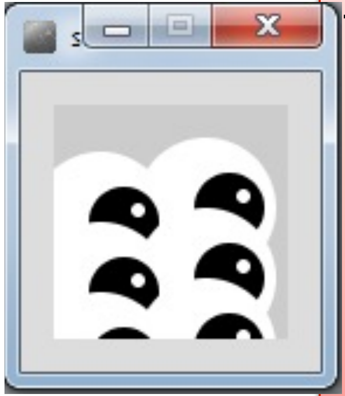
```
void eye (int x, int y)
{
   fill(255);
   ellipse(x,y,60,60); //outer white circle
   fill(0);
   ellipse(x+10, y, 30, 30); //black circle
   fill(255);
   ellipse(x+16, y-5, 6, 6); //small, white circle
}
```

Example 6.3 – Drawing six eyes

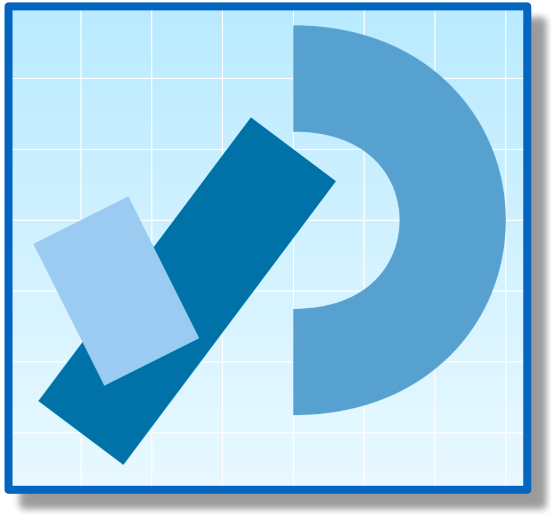
```
void setup()
{
    size(100,100);
    noStroke();
}
```

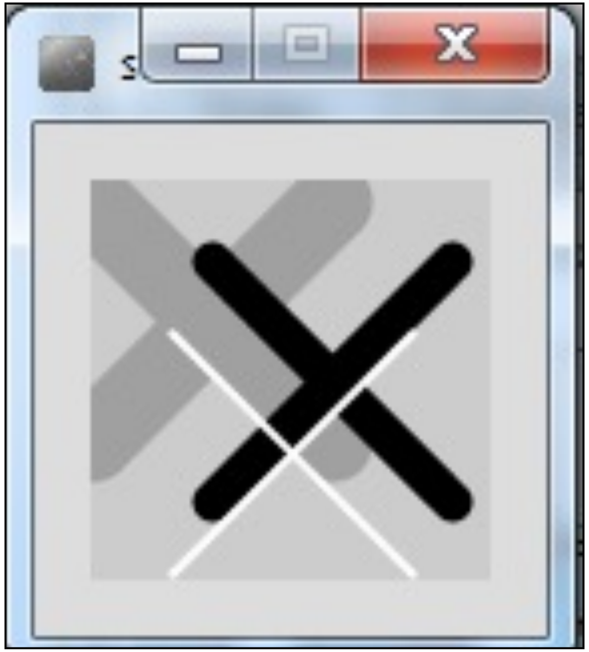
```
void eye (int x, int y)
{
    fill(255);
    ellipse(x,y,60,60); //oute
    fill(0);
    ellipse(x+10, y, 30, 30); //black circle
    fill(255);
    ellipse(x+16, y-5, 6, 6); //small, white circle
}
```

```
void draw()
{
    background(204);
    eye(65,44);
    eye(20,50);
    eye(65,74);
    eye(20,80);
    eye(65,104);
    eye(20,110);
}
```



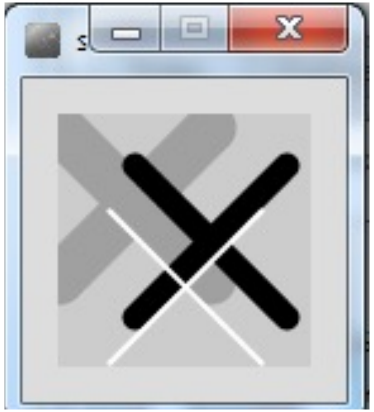
Method example: X's





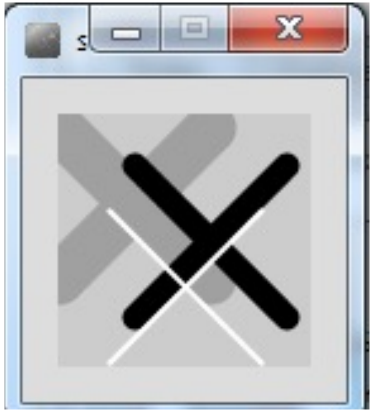
How about this solution?

```
void setup() {
  size(100,100);
}
```



```
void draw(){
  background(204);
  //draw thick, light gray x
  stroke(160);
  strokeWeight(20);
  line(0,5,60,65);
  line(60,5,0,65);
  //draw medium, black x
  stroke(0);
  strokeWeight(10);
  line(30,20,90,80);
  line(90,20,30,80);
  //draw thin, white x
  stroke(255);
  strokeWeight(2);
  line(20,38,80,98);
  line(80,38,20,98);
}
```

Code duplication



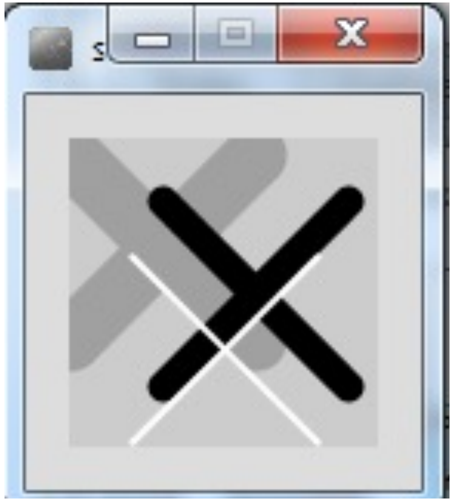
```
//draw thick, light gray x
  stroke(160);
  strokeWeight(20);
  line(0,5,60,65);
  line(60,5,0,65);
```

```
//draw medium, black x
  stroke(0);
  strokeWeight(10);
  line(30,20,90,80);
  line(90,20,30,80);
```

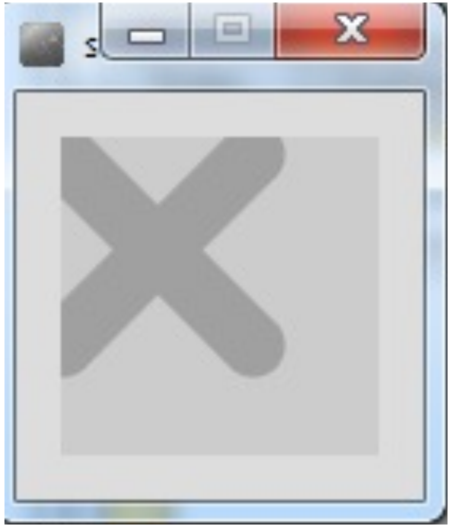
```
//draw thin, white x
  stroke(255);
  strokeWeight(2);
  line(20,38,80,98);
  line(80,38,20,98);
```

A solution with methods

- We will incrementally build a solution that uses methods to produce this output…



Example 6.4 – a method to draw a thick, light gray X

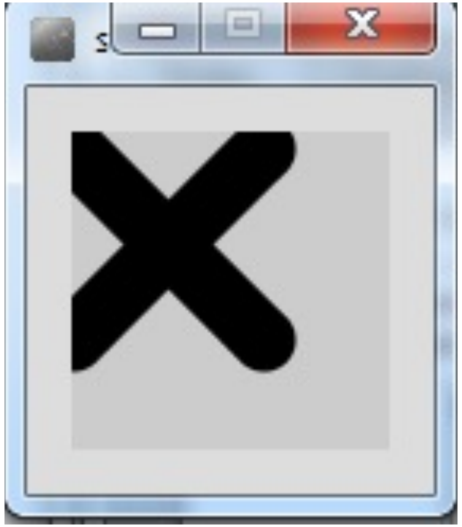


```
void draw()
{
    background(204);
    drawX();
}
```

```
void drawX()
{

   stroke(160);
   strokeWeight(20);
   line(0,5,60,65);
   line(60,5,0,65);
}
```

Example 6.5 – drawing a thick X, passing colour as a parameter

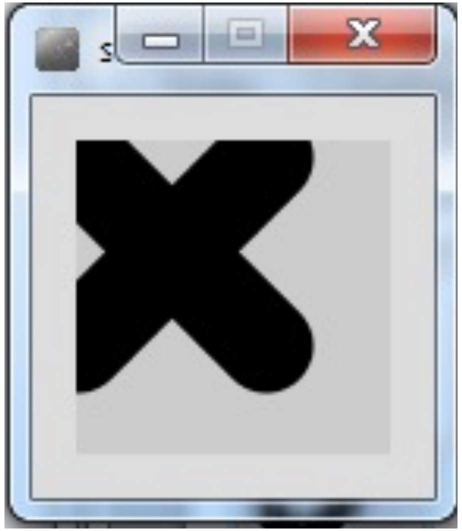


```
void draw()
{
    background(204);
    drawX(0);
}
```

```
void drawX (int gray)
{

  stroke(gray);
  strokeWeight(20);
  line(0,5,60,65);
  line(60,5,0,65);
}
```

Example 6.6 – drawing X, passing colour and weight.

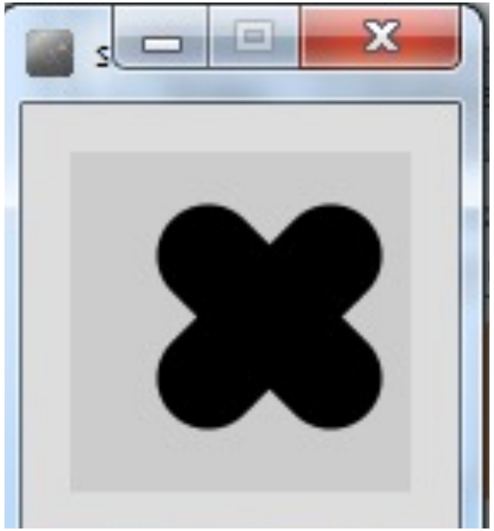


```
void draw()
{
    background(204);
    drawX(0, 30);
}
```

```
void drawX (int gray, int weight)
{

  stroke(gray);
  strokeWeight(weight);
  line(0,5,60,65);
  line(60,5,0,65);
}
```

Example 6.7 – drawing X, passing colour, weight, position, size



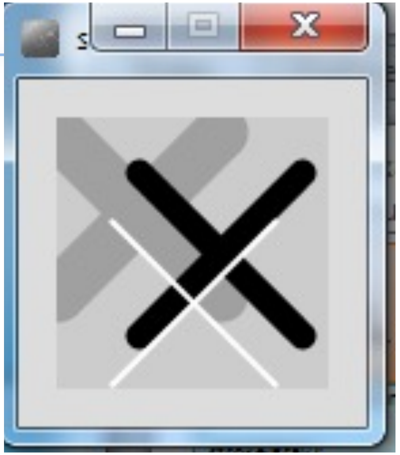
```
void draw()
{
    background(204);
    drawX(0, 30, 40, 30, 36);
}
```

```
void drawX (int gray, int weight,
int x, int y, int size)
{
  stroke(gray);
  strokeWeight(weight);
  line(x, y, x+size, y+size);
  line(x+size, y, x, y+size);
}
```

Example 6.8 – drawing multiple Xs

```
void draw()
{
      background(204);
      drawX(160, 20, 0, 5, 60);
      drawX(0, 10, 30, 20, 60);
      drawX(255, 2, 20, 38, 60);
}
```

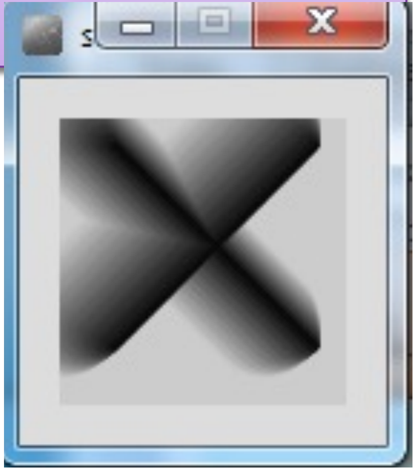
```
void drawX (int gray, int weight, int x, int y, int size)
{
  stroke(gray);
  strokeWeight(weight);
  line(x, y, x+size, y+size);
  line(x+size, y, x, y+size);
}
```



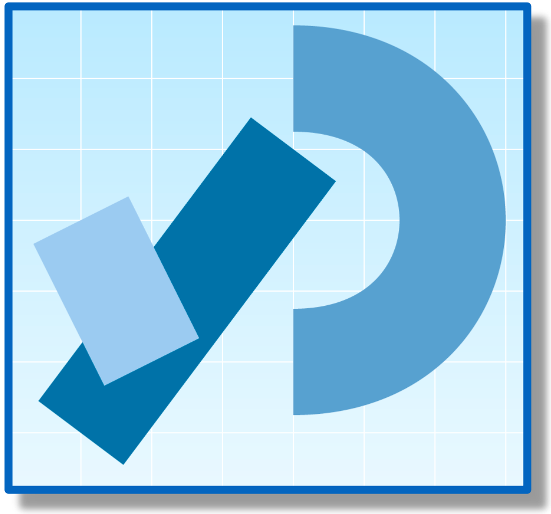
Example 6.9 – drawing multiple Xs using a for loop

```
void draw()
{
        background(204);
        for (int i = 0; i < 20; i++){
                drawX(200-i*10, (20-i)*2, i, i/2, 70);
        }
}
```

```
void drawX(int gray, int weight, int x,
int y, int size)
{
  stroke(gray);
  strokeWeight(weight);
  line(x, y, x+size, y+size);
  line(x+size, y, x, y+size);
}
```



Overloading Methods



Overloaded methods

- Multiple methods can have the same name, once they have a different parameter list.
- In the previous examples, we wrote the following methods:
  - `void drawX ()`
  - `void drawX (int gray)`
  - `void drawX (int gray, int weight)`
  - `void drawX (int gray, int weight, int x, int y, int size)`

Overloaded methods

- Multiple methods can have the same name, once they have a different parameter list.
- In the previous examples, we wrote the following methods:

```
void drawX ()
void drawX (int gray)
void drawX (int gray, int weight)
void drawX (int gray, int weight, int x, int y, int size)
```

Same Name

Different Parameter List

Overloaded methods

Method signature	Parameter List
void drawX ()	no parameter
void drawX (int gray)	int
void drawX (int gray, int weight)	int, int
void drawX (int gray, int weight, int x, int y, int size)	int, int, int, int, int

Overloaded methods

- A program can have two or more methods with the same name, only if their parameter list is different.

- When Java is checking that a parameter list is different, it is not checking the name of the variables, it is checking the data type of the variables
  e.g. this is permitted as the data type is different:

```
void drawX ( int    gray)
void drawX ( float  gray)
```

Data types must be different

Overloaded methods

```
void draw()
{
    background(204);
   drawX(0);
}
```

Which **drawX** method is called and why?

```
void drawX(int gray){
   stroke(gray);
   strokeWeight(5);
   line(0,5,60,65);
   line(60,5,0,65);
}

void drawX(float gray){
   stroke(gray);
   strokeWeight(20);
   line(0,5,60,65);
   line(60,5,0,65);
}
```

Overloaded methods

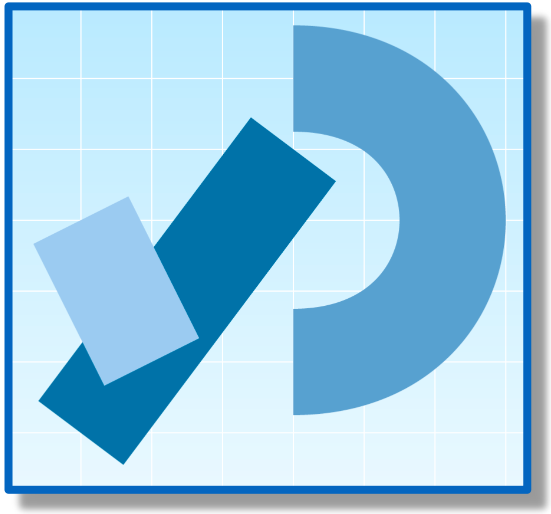
- When you call a method, Java **matches** the **number and type of the arguments** you passed to the method with all the declared methods.
- When a match is found, Java invokes that method

  e.g.

  `drawX(0)` calls `void drawX (int gray)`

  `drawX(0.0)` calls `void drawX (float gray)`

Method example:
Celcius / Farenheit Converter



Example 6.10 – Farenheit to Celsius

```
void setup()
{
    float celsius = farenheitToCelsius (451.0);
    println("Celsius value is: " + celsius);
}
```

Farenheit value is hardcoded as a literal

```
float farenheitToCelsius (float farenheit)
{
    float result = (farenheit - 32.0) * (5.0/9.0);
    return result;
}
```

Return type

Example 6.10 – Farenheit to Celsius

Farenheit value is hardcoded as a literal

```
void setup()
{
    float celsius = farenheitToCelsius (451.0);
    println("Celsius value is: " + celsius);
}
```

```
Celsius value is: 232.77779
```

```
float farenheitToCelsius (float farenheit)
{
    float result = (farenheit - 32.0) * (5.0/9.0);
    return result;
}
```

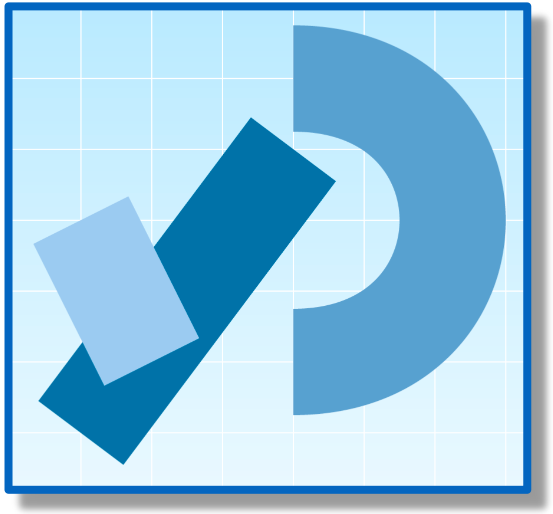
Example 6.10 – Updated

```
float farenheitToCelsius (float farenheit)
{
     float result = (farenheit - 32.0) * (5.0/9.0);
     return result;
}
```

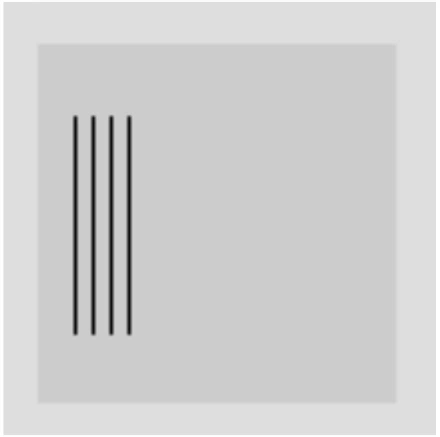
both methods are exactly the same

```
float farenheitToCelsius (float farenheit)
{
     return (farenheit - 32.0) * (5.0/9.0);
}
```

Recursion



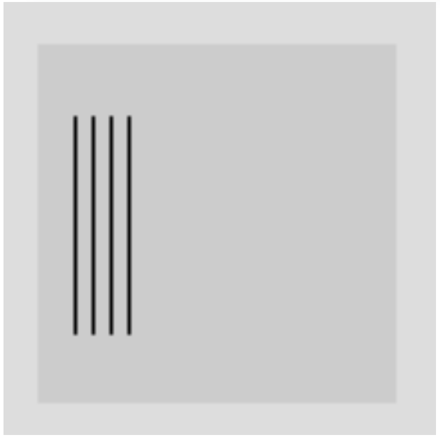
Example 6.11 – drawLines – for loop



```
void setup()
{
  size(100,100);
  drawLines(10,4);
}
```

```
void drawLines (int xStart, int numLines)
{
    for (int i = 0; i < numLines; numLines--)
    {
        line (xStart, 20, xStart, 80);
        xStart += 5;
    }
}
```

Example 6.11 – drawLines – for loop



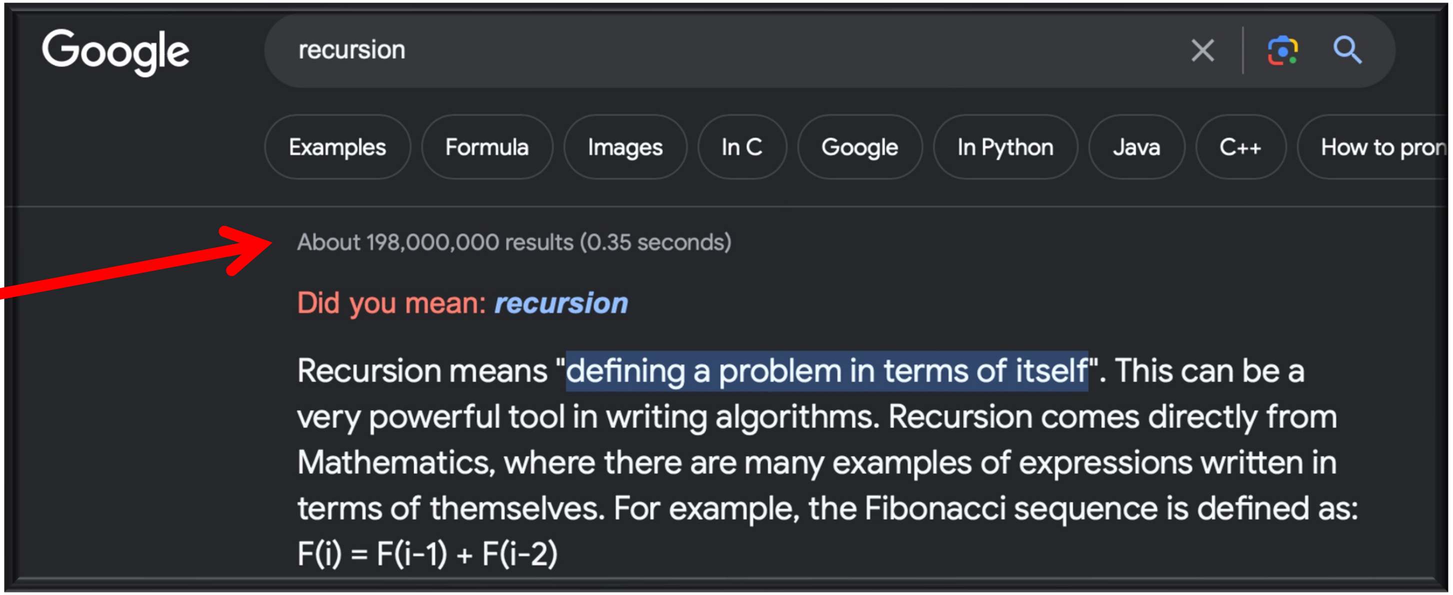
```
void setup()
{
   size(100,100);
   drawLines(10,4);
}
```

```
void drawLines (int xStart, int numLines)
{
     for (int i = 0; i < numLines; numLines--)
     {
          line (xStart, 20, xStart, 80);
          xStart += 5;
     }
}
```

NOTE
instead of incrementing the loop control variable i as normal (e.g. i++)
the condition is being reduced back to 0 (by decrementing `numLines`)

Recursion

- A method can contain a **line of code** that calls itself
  - This is called recursion.


Google

recursion

Examples | Formula | Images | In C | Google | In Python | Java | C++ | How to pron

About 198,000,000 results (0.35 seconds)

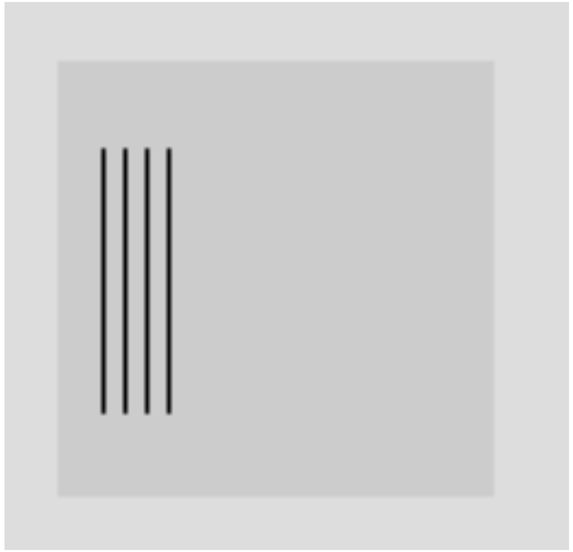
Did you mean: ***recursion***

Recursion means "defining a problem in terms of itself". This can be a very powerful tool in writing algorithms. Recursion comes directly from Mathematics, where there are many examples of expressions written in terms of themselves. For example, the Fibonacci sequence is defined as: F(i) = F(i-1) + F(i-2)

Recursion continued…

- To stop the infinite calling of the method, it is necessary to have some way for the method to exit.
    - This is called the *base case*.
    - You continually work towards the base case.

Example 6.11 – drawLines – recursion

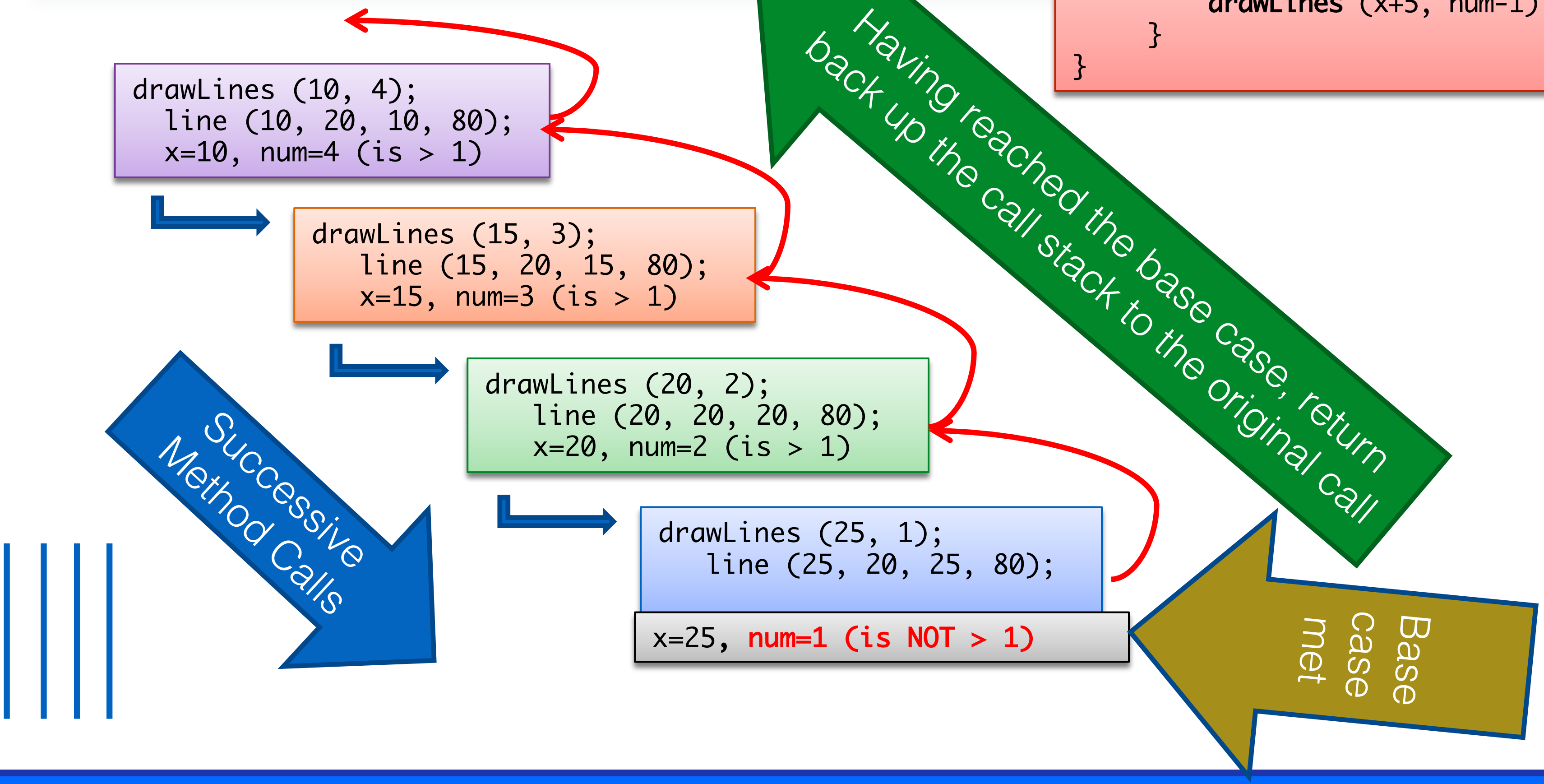


```
void setup()
{
  size(100,100);
  drawLines(10,4);
}
```

```
void drawLines (int x, int num)
{
     line (x, 20, x, 80);
     if (num > 1)
     {
          drawLines (x+5, num-1);
     }
}
```

Example 6.11

```
void drawLines (int x, int num){
     line (x, 20, x, 80);
     if (num > 1)
     {
          drawLines (x+5, num-1);
     }
}
```



```
drawLines (10, 4);
  line (10, 20, 10, 80);
  x=10, num=4 (is > 1)
```

```
drawLines (15, 3);
   line (15, 20, 15, 80);
   x=15, num=3 (is > 1)
```

```
drawLines (20, 2);
   line (20, 20, 20, 80);
   x=20, num=2 (is > 1)
```

```
drawLines (25, 1);
   line (25, 20, 25, 80);
x=25, num=1 (is NOT > 1)
```

Successive Method Calls
Having reached the base case, return back up the call stack to the original call
Base case met

Summary

1. Method example: Eyes
2. Method example: X’s
3. Overloading methods
4. Method example: Celcius / Farenheit Converter
5. Recursion

Questions?

References

- Reas, C. & Fry, B. (2014) Processing – A Programming Handbook for Visual Designers and Artists, 2nd Edition, MIT Press, London.

Thanks.