Using Methods

Writing your own methods

Produced by: Dr. Siobhán Drohan
Mr. Colm Dunphy
Mr. Diarmuid O'Connor

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Department of Computing and Mathematics
http://www.wit.ie/

Topics list

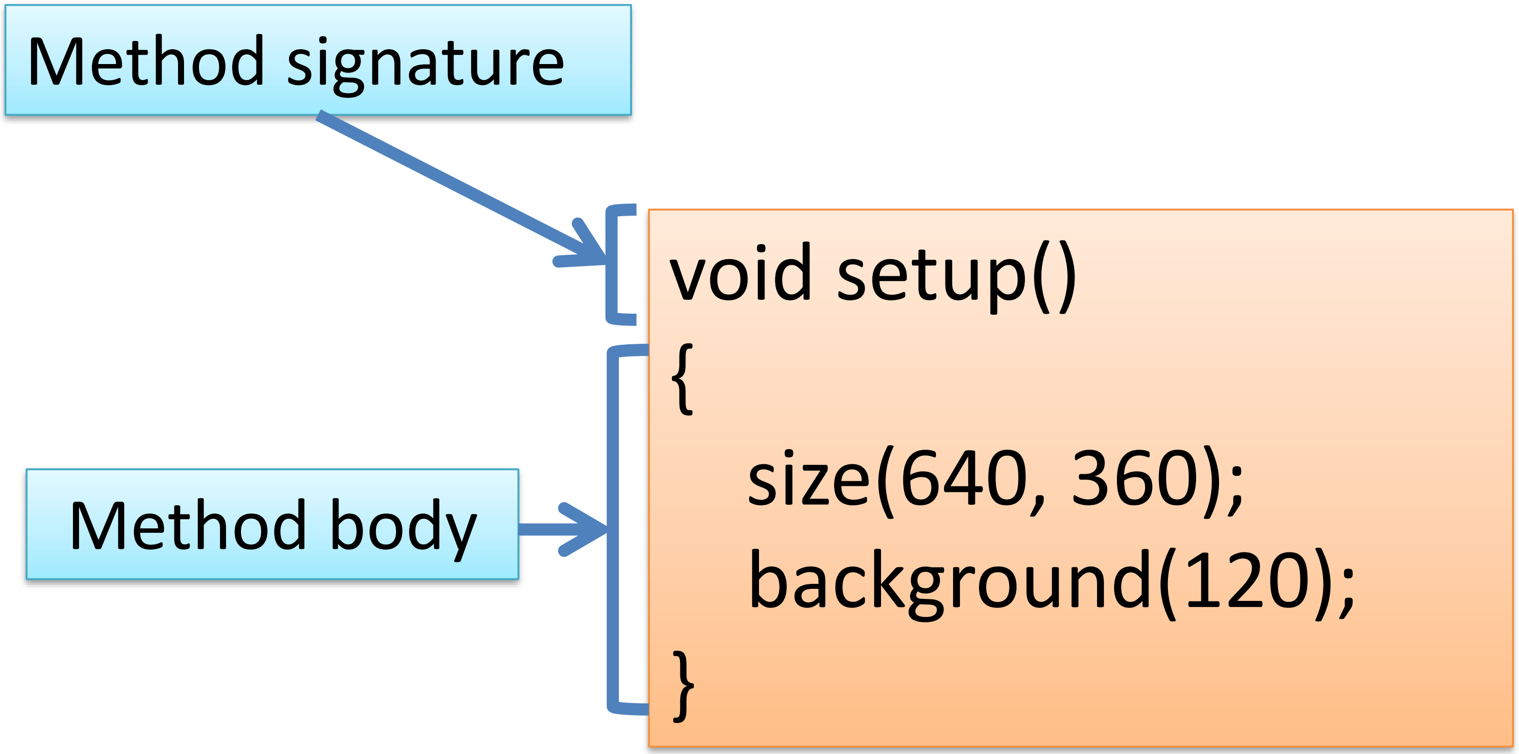
1. Recap of method **terminology**:
   - Return type
   - Method names
   - Parameter list

2. **Writing your own** methods:
   - With no parameters
   - With parameters
   - That return data

Recap: Methods in Processing

- A method comprises a **set of instructions that performs some task**.
- When we **invoke** the method,
  it performs the task.
- Some methods that we have used are:
  – rect, ellipse, stroke, line, fill, etc.
  – void mousePressed()
  – void setup, void draw()

Recap: Method terminology

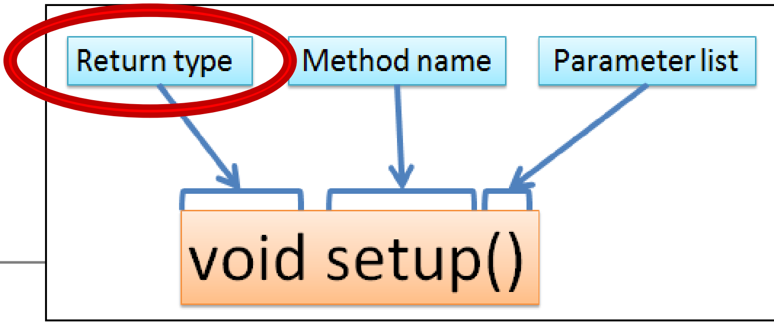


```
void setup()
{
    size(640, 360);
    background(120);
}
```

Recap: Method **signature**

$$\overbrace{\texttt{void}}^{\text{Return type}}\ \overbrace{\texttt{setup}}^{\text{Method name}}\overbrace{\texttt{()}}^{\text{Parameter list}}$$

Recap: Return Types



- Methods can **return information**.
- The void keyword means that **nothing** is returned from the method.
- When a **data type** (e.g. int) appears before the method name, this means that something is returned from the method.
- Within the body of the method, you use the **return** statement to return the value.
- You can **only have one return type per method**.
- Methods can return any type of data e.g. boolean, byte, char, int, float, String, etc.

Recap: Return Types

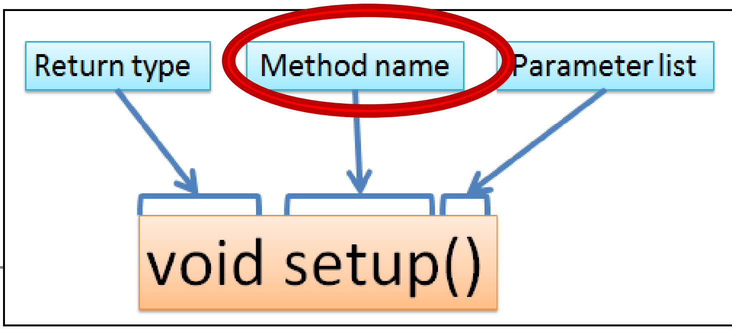
```
int val = 30;


void draw()
{
    int result = timestwo(val);
    println(result);
}
```

```
int timestwo(int number)
{
   number = number * 2;
   return number;
}
```

// The red **int** in the function declaration
// specifies the type of data to be returned.

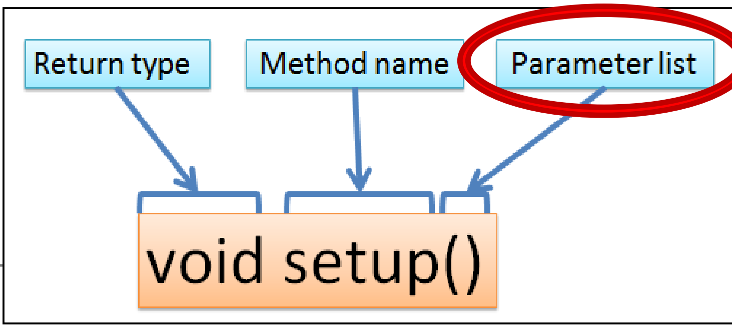
https://processing.org/reference/return.html

Recap: Method name



- Method names should:
  - Use **verbs** (i.e. actions) to describe what the method does e.g.
    - calculateTax
    - printResults
  - Be **mixed case (camelCase)** with the first letter lowercase and the first letter of each subsequent internal word capitalised.

Recap: Parameter list



- Methods take in data via their **parameters**.

Methods do not have to pass parameters.

These methods don't need any additional information to do their tasks.

If a method needs additional information to execute, we provide a parameter so that the information can be passed into it.

A method can have any number of parameters.

void noStroke()
void setup()
void noCursor()

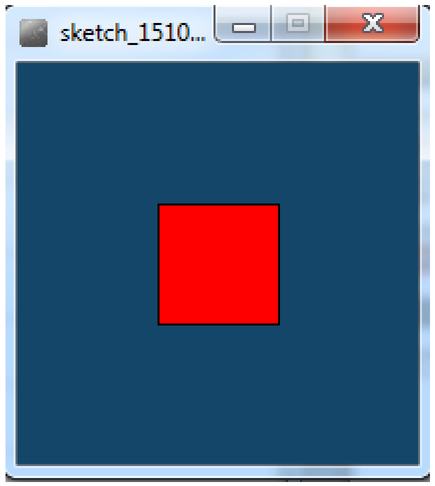
void strokeWeight (float weight)
void size (int width, int height)

Topics list

1. Recap of method **terminology**:
   - Return type
   - Method names
   - Parameter list

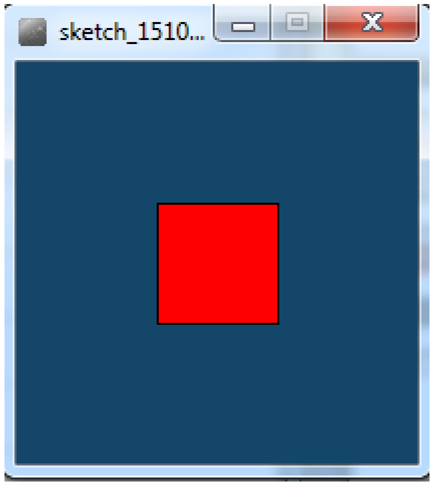
2. **Writing your own** methods:
   - ➡ With no parameters
   - With parameters
   - That return data

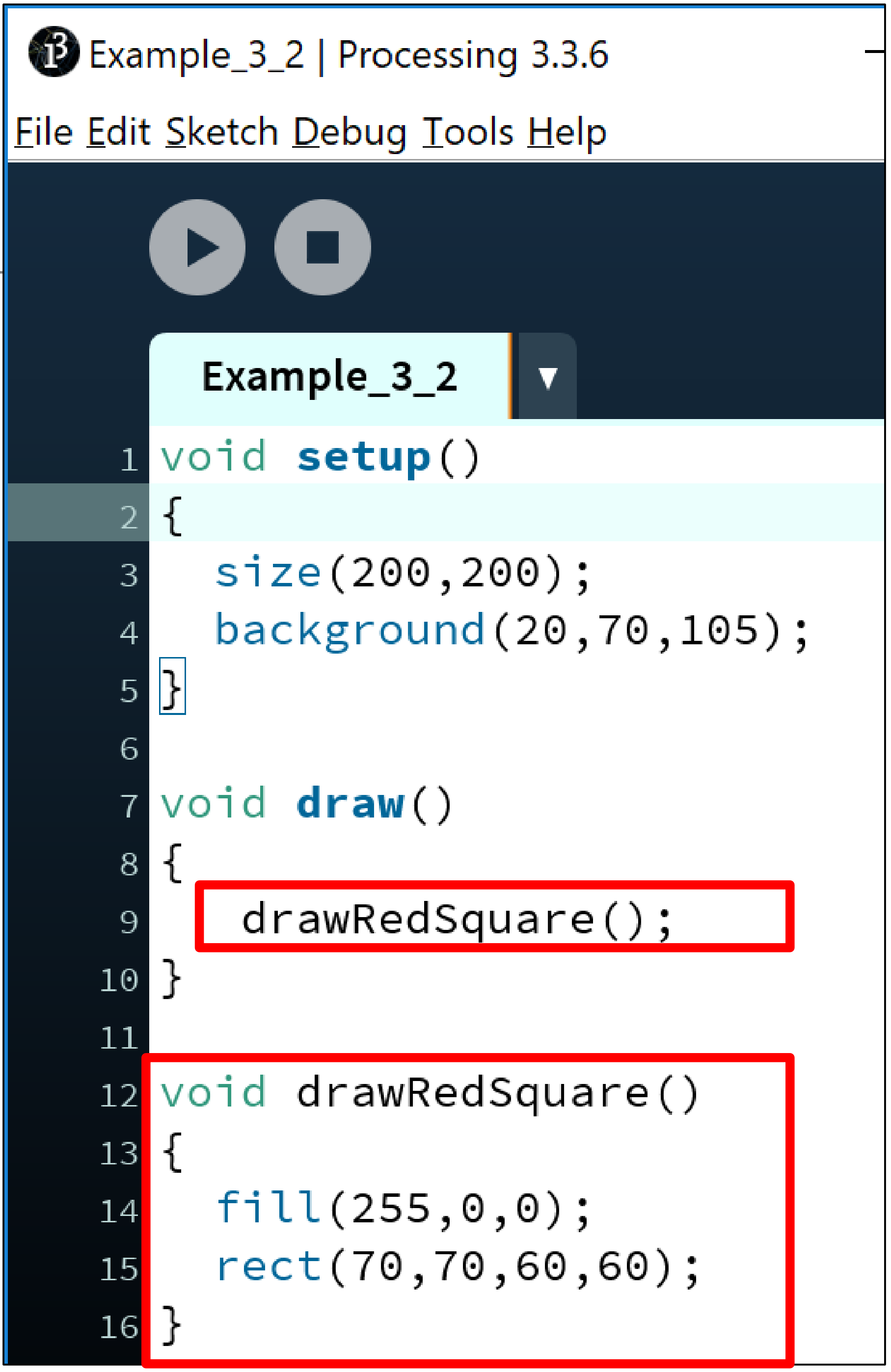
Writing methods **with NO parameters**



- Draw a red square at certain (x, y) coordinates.

Processing
Example 3.2




Example_3_2 | Processing 3.3.6
File Edit Sketch Debug Tools Help
Example_3_2

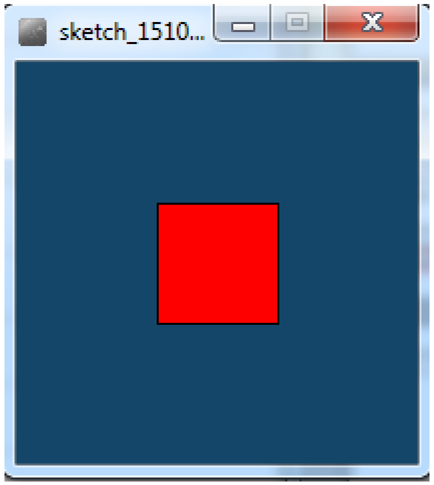
```
void setup()
{
  size(200,200);
  background(20,70,105);
}

void draw()
{
  drawRedSquare();
}

void drawRedSquare()
{
  fill(255,0,0);
  rect(70,70,60,60);
}
```

Topics list

1. Recap of method **terminology**:
   - Return type
   - Method names
   - Parameter list

2. **Writing your own** methods:
   - With no parameters
   - With parameters
   - That return data

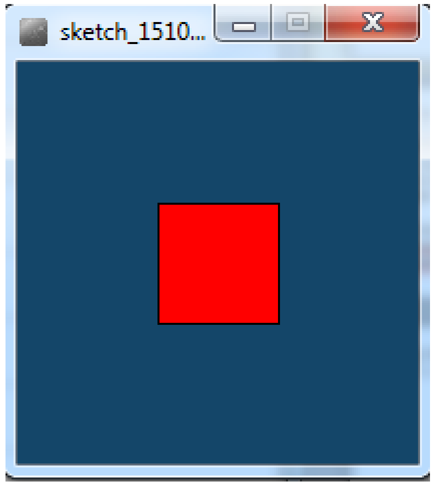
Writing methods **with parameters**

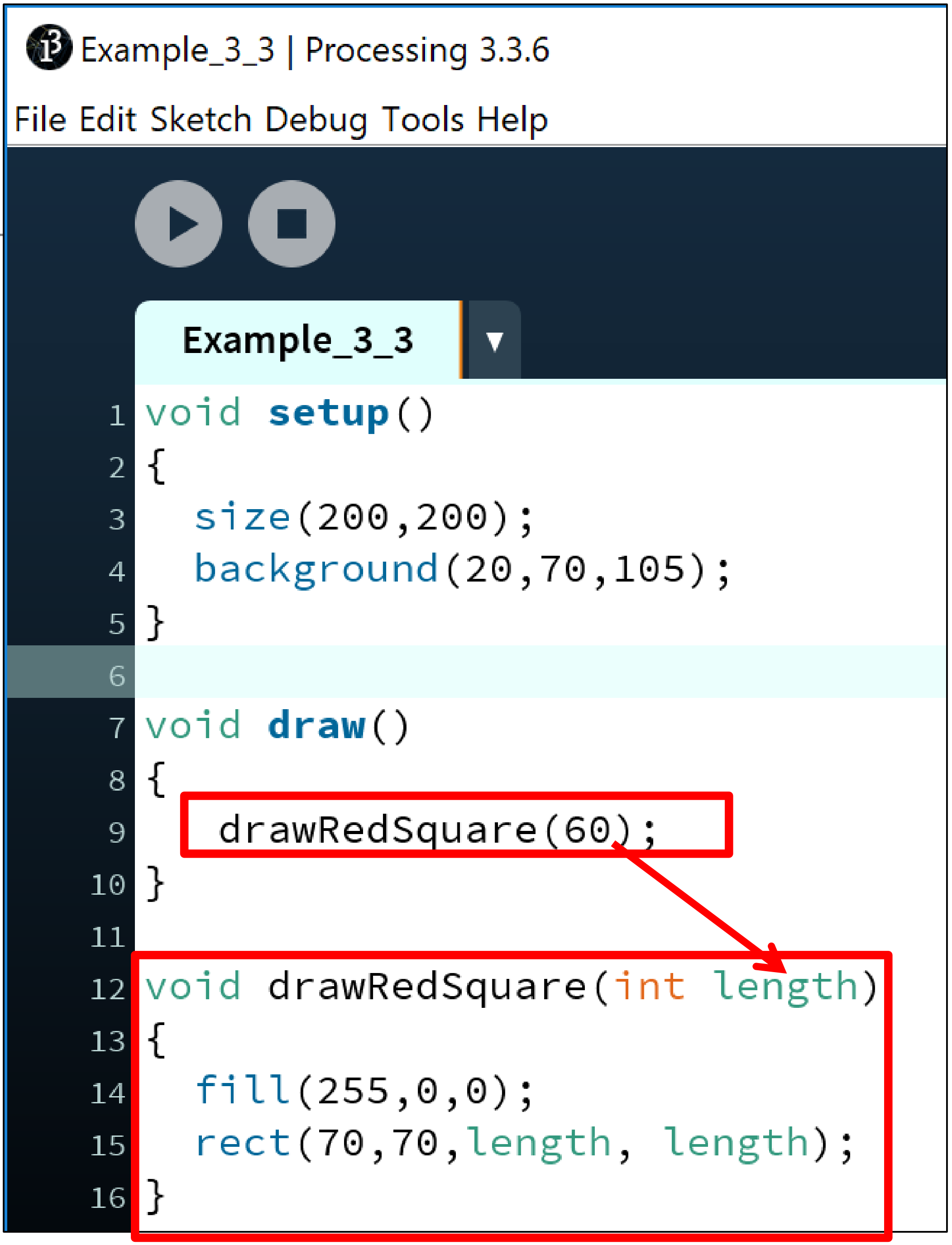


- Now update the code so that you can:

  pass in the length
  of the square
  into the method,
  drawRedSquare.

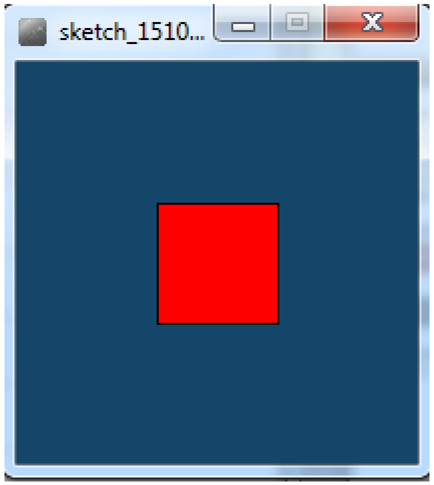
Processing
Example 3.3




Example_3_3 | Processing 3.3.6
File Edit Sketch Debug Tools Help
Example_3_3

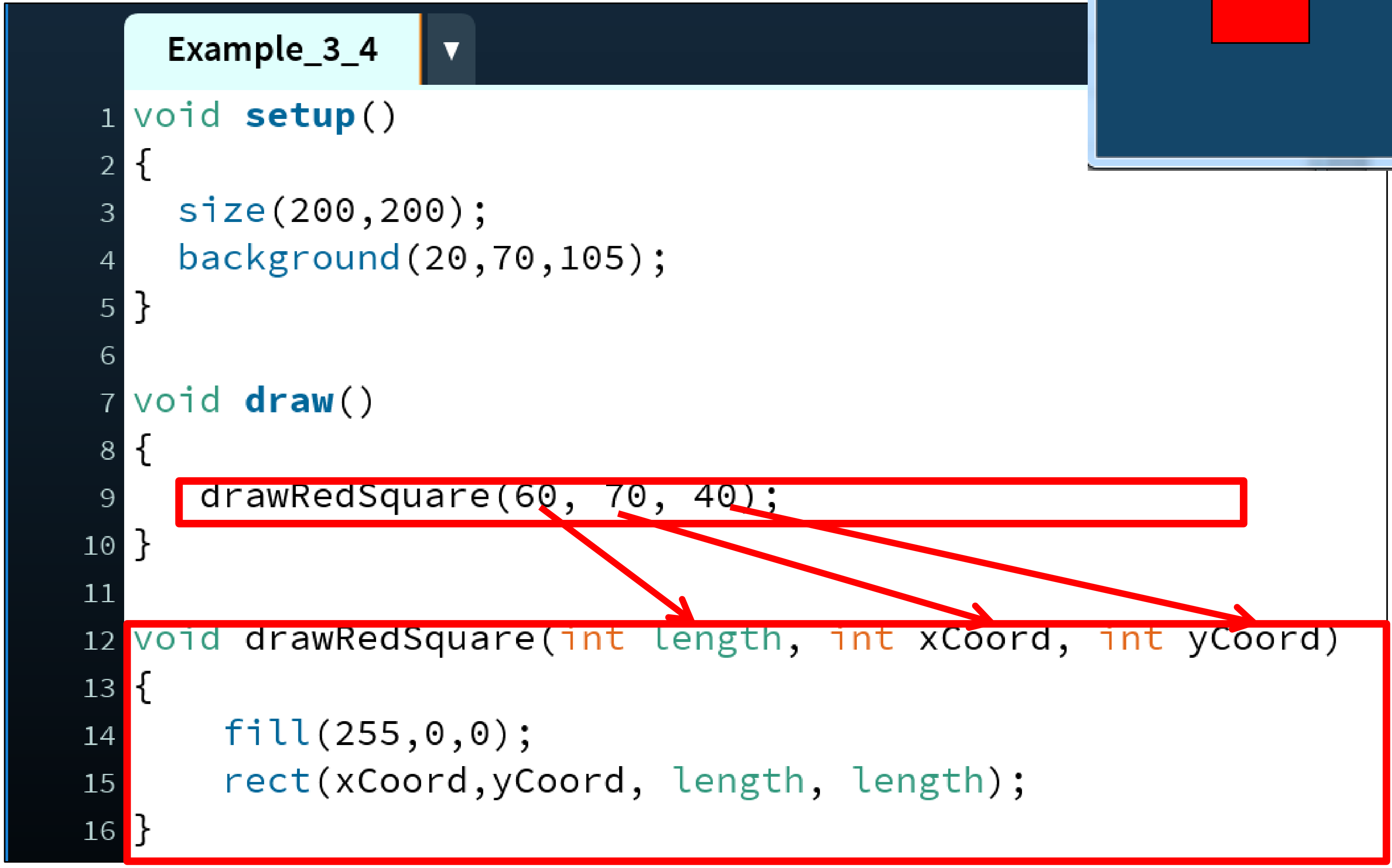
```
void setup()
{
  size(200,200);
  background(20,70,105);
}

void draw()
{
  drawRedSquare(60);
}

void drawRedSquare(int length)
{
  fill(255,0,0);
  rect(70,70,length, length);
}
```

Writing methods **with parameters**

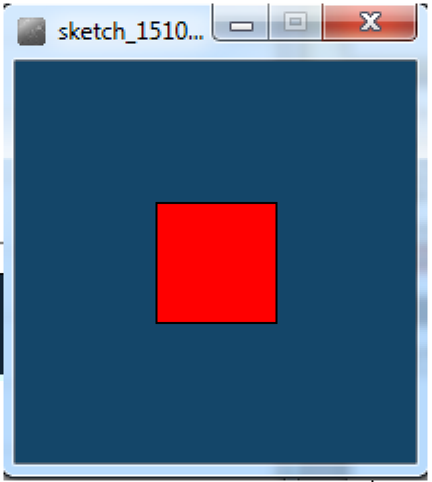


- Now update the code so that you can pass in the:
  - **length** of the square
  - **xCoordinate** of the square
  - **yCoordinate** of the square
- into the method, drawRedSquare.

Processing
Example 3.4

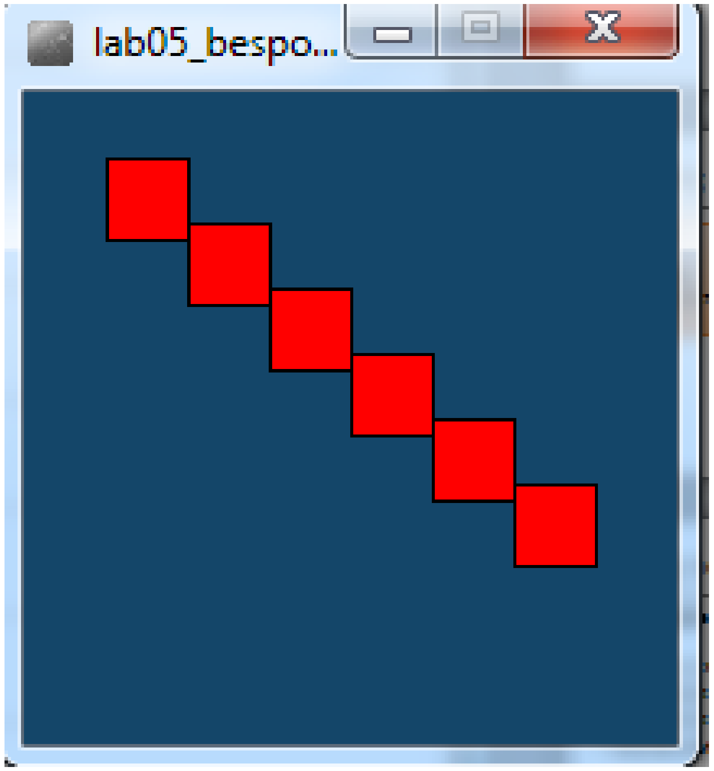


```
void setup()
{
  size(200,200);
  background(20,70,105);
}

void draw()
{
   drawRedSquare(60, 70, 40);
}

void drawRedSquare(int length, int xCoord, int yCoord)
{
     fill(255,0,0);
     rect(xCoord,yCoord, length, length);
}
```

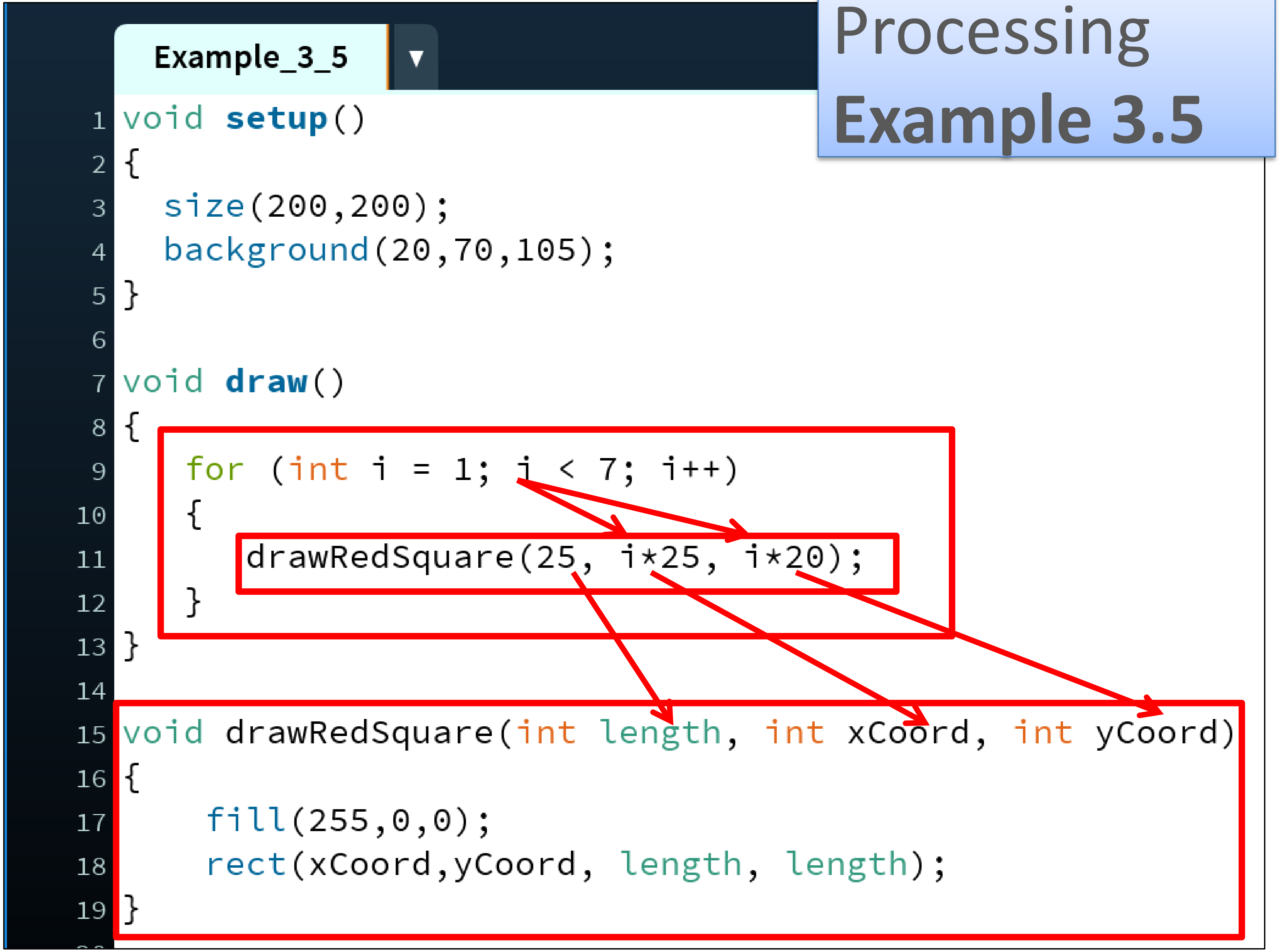


Writing methods with parameters

- Now update the code so that you can call the drawRedSquare multiple times (using a loop).



Processing **Example 3.5**



```
void setup()
{
  size(200,200);
  background(20,70,105);
}

void draw()
{
   for (int i = 1; i < 7; i++)
   {
      drawRedSquare(25, i*25, i*20);
   }
}

void drawRedSquare(int length, int xCoord, int yCoord)
{
    fill(255,0,0);
    rect(xCoord,yCoord, length, length);
}
```

Topics list

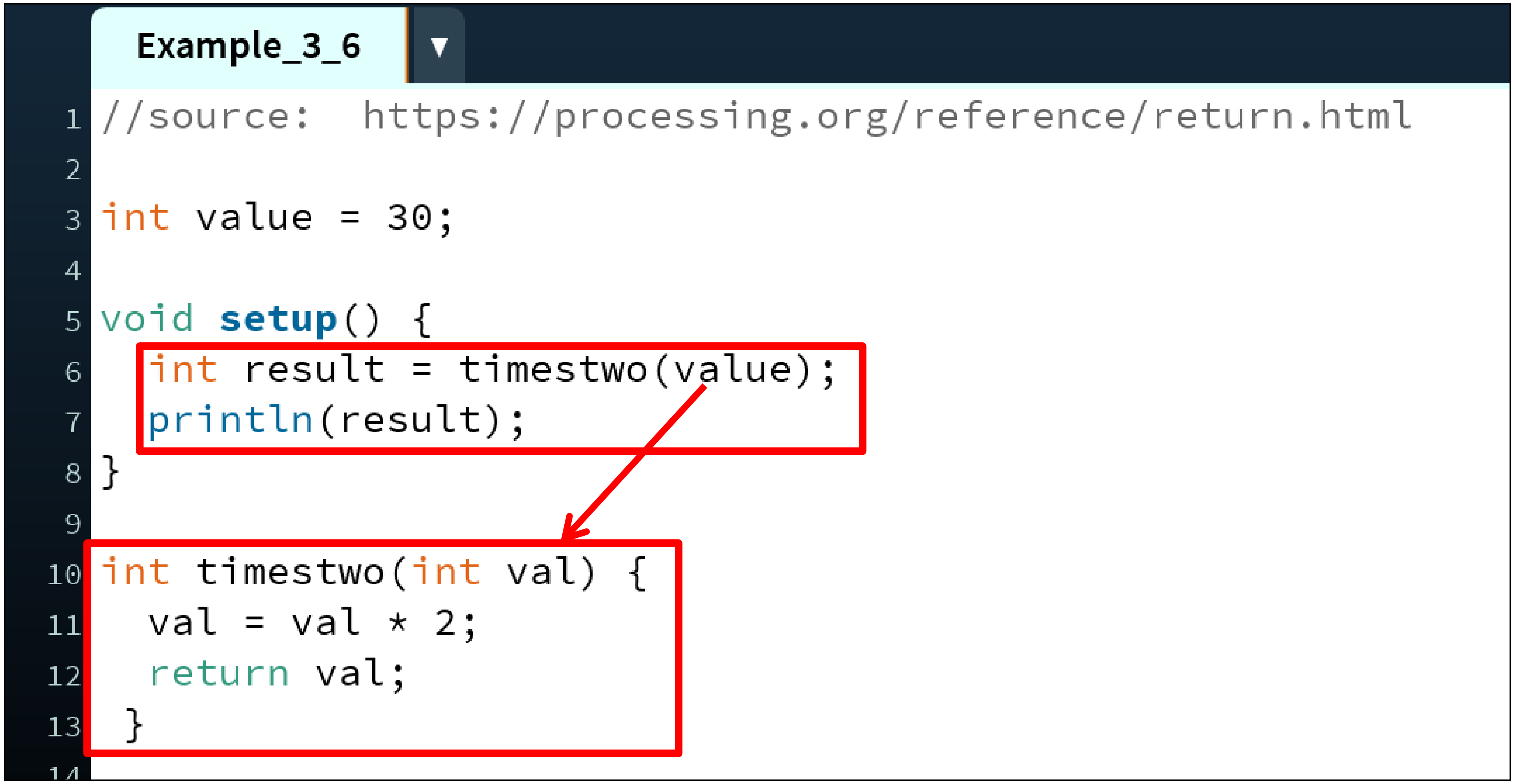
1. Recap of method **terminology**:
   - Return type
   - Method names
   - Parameter list

2. **Writing your own** methods:
   - With no parameters
   - With parameters
   - ➡ That return data

Writing methods **that return data**

- Write a method called **timesTwo.**
- This method should
  - **take in one int parameter**.
  - **multiply this int by 2** and
  - **return** it back to where the **timesTwo** method was called from.
  - The returned value should be **printed to the console**.

Processing **Example 3.6**



```
//source:  https://processing.org/reference/return.html

int value = 30;

void setup() {
  int result = timestwo(value);
  println(result);
}

int timestwo(int val) {
  val = val * 2;
  return val;
 }
```

Summary

1. Recap of method **terminology**:
   - Return type
   - Method names
   - Parameter list

2. **Writing your own** methods:
   - With no parameters
   - With parameters
   - That return data

Questions?