

Scope of variables, Printing and Compound Assignment Statements

Produced Dr. Siobhán Drohan
by: 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. Use of `println()`, `text()` in Processing**
- 2. Variable Scope**
- 3. Compound Assignment Statements**

println() and **text()** in Processing

- To print a message to the **console** in Processing, use:
 - **print()**
 - **println()**
- Both take a String as input,
 - (more on this in later lectures).
- To print onto the **display window**, use:
 - **text()**

File Edit Sketch Debug Tools Help



Java ▾

sketch_180122a ▾

```
1 print("Hello ");
2 println("there");
3
4 println("This is advancing the cursor onto the next line");
5 println("And this is also advancing the cursor to the next line");
6
7
8
9
10
11
```

Hello there
This is advancing the cursor onto the next line
And this is also advancing the cursor to the next line

Console

Errors

println()

Each statement prints the same output.

The screenshot shows the Processing IDE interface. At the top, it says "sketch_180122a | Processing 3.3.6". The menu bar includes File, Edit, Sketch, Debug, Tools, and Help. Below the menu is a toolbar with play and stop buttons, and a "Java" dropdown. The code editor window is titled "sketch_180122a". It contains the following code:

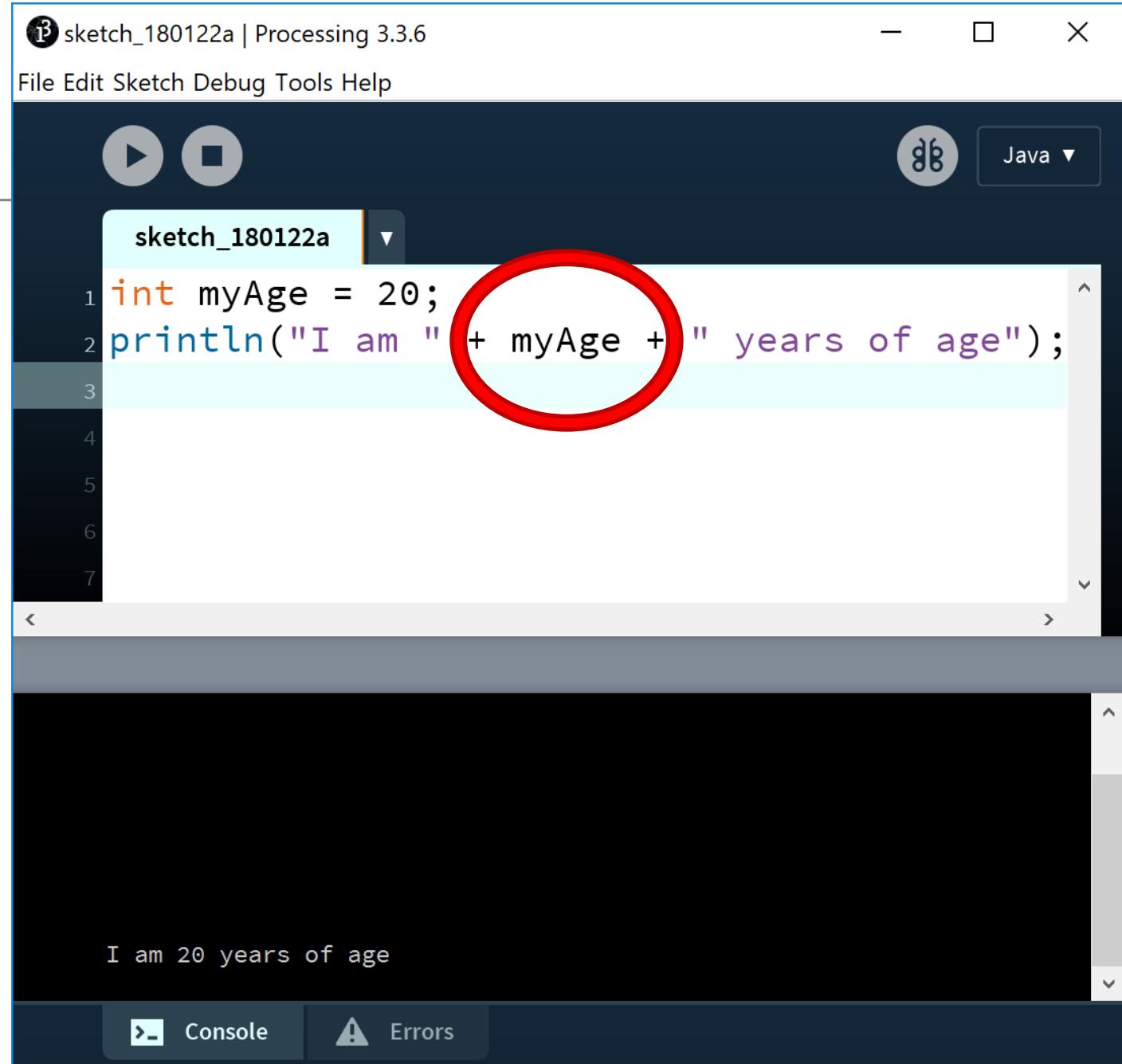
```
1 println("Hello World");
2 println("Hello " + "World");
3 println("Hell" + "o World");
```

The code editor has line numbers 1 through 7 on the left. The third line is highlighted with a light blue background. Below the code editor is a large black area representing the canvas. At the bottom, there are two tabs: "Console" and "Errors". The "Console" tab is active, showing the output of the code:

```
Hello World
Hello World
Hello World
```

println()

We can use variables in the print statement.



The screenshot shows the Processing IDE interface with the title bar "sketch_180122a | Processing 3.3.6". The menu bar includes File, Edit, Sketch, Debug, Tools, and Help. Below the menu is a toolbar with play and stop buttons, and a "Java" dropdown. The code editor window displays the following Java code:

```
int myAge = 20;
println("I am " + myAge + " years of age");
```

A red circle highlights the string concatenation operator "+ myAge +". The output window at the bottom shows the result of the println statement: "I am 20 years of age". At the bottom of the screen are tabs for "Console" and "Errors".

text() in processing

- **text()** is used to draw text on the display window.

```
textSize(32);  
text("word", 10, 30);  
fill(0, 102, 153);  
text("word", 10, 60);  
fill(0, 102, 153, 51);  
text("word", 10, 90 );
```



Text to be written
(also in String format)

x, y co-ordinates on screen

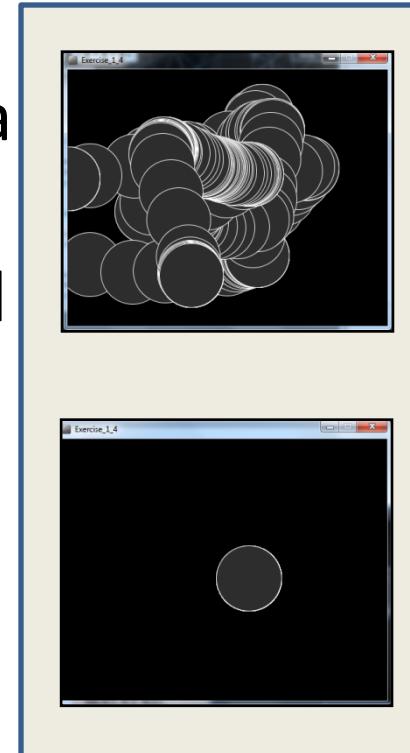
Topics list

1. Use of **println()**, **text()** in Processing
2. Variable **Scope**
3. Compound Assignment Statements

Recap: Processing Example 2.8

Functionality:

- Draw a circle on the mouse (x,y) coordinates.
- Each time you move the mouse, draw a new circle.
- All the circles remain in the sketch until you press a mouse button.
- When you press a mouse button, the sketch is cleared and a single circle is drawn at the mouse (x,y) coordinates.



Recap: Processing Example 2.8

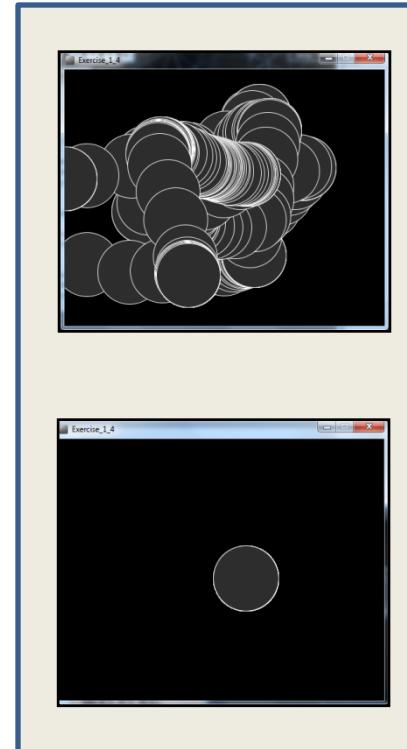
Example_2_8 | Processing 3.3.6

File Edit Sketch Debug Tools Help

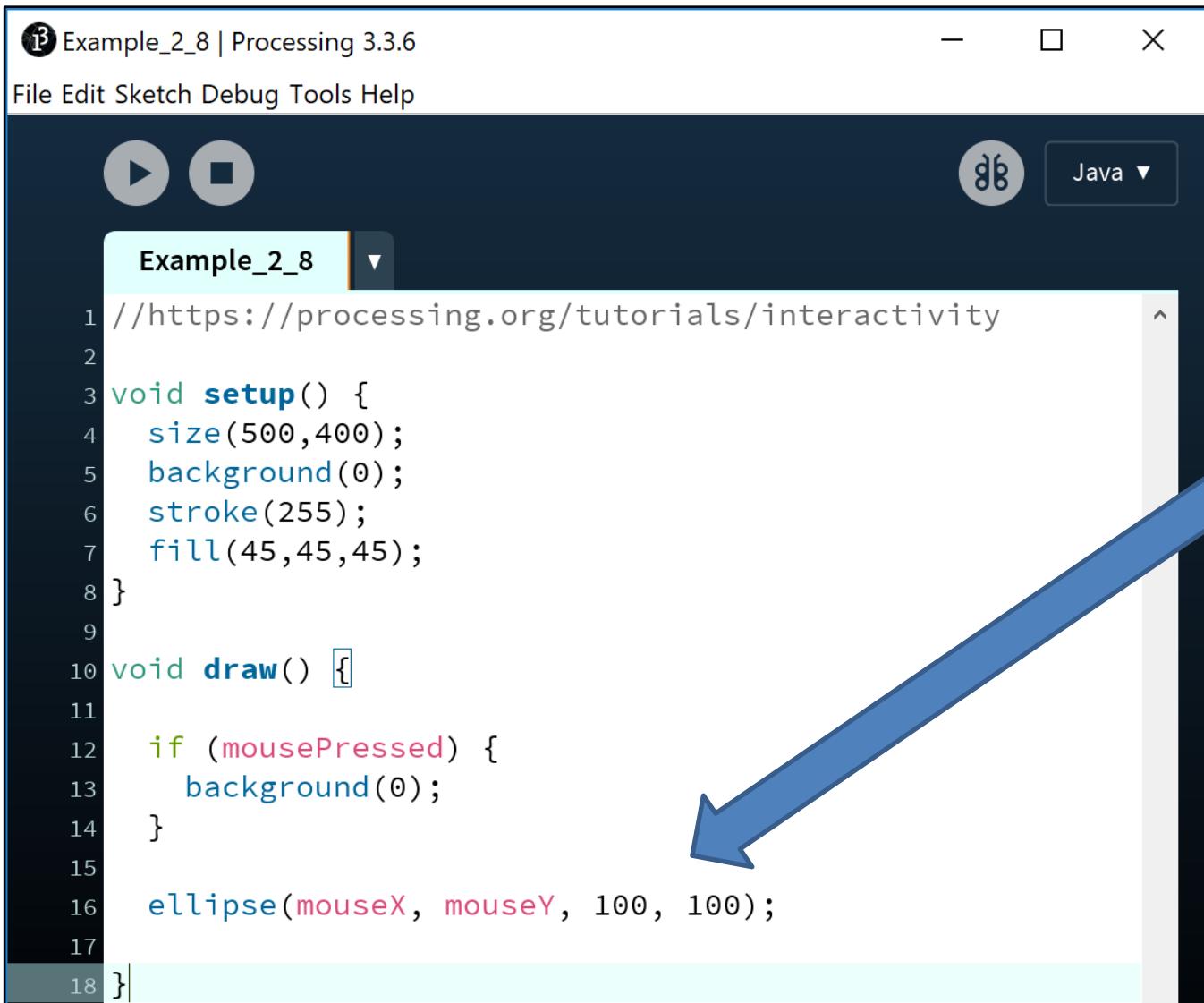
Java ▾

Example_2_8

```
1 //https://processing.org/tutorials/interactivity
2
3 void setup() {
4     size(500,400);
5     background(0);
6     stroke(255);
7     fill(45,45,45);
8 }
9
10 void draw() {
11
12     if (mousePressed) {
13         background(0);
14     }
15
16     ellipse(mouseX, mouseY, 100, 100);
17
18 }
```



Recap: Processing Example 2.8



```
//https://processing.org/tutorials/interactivity

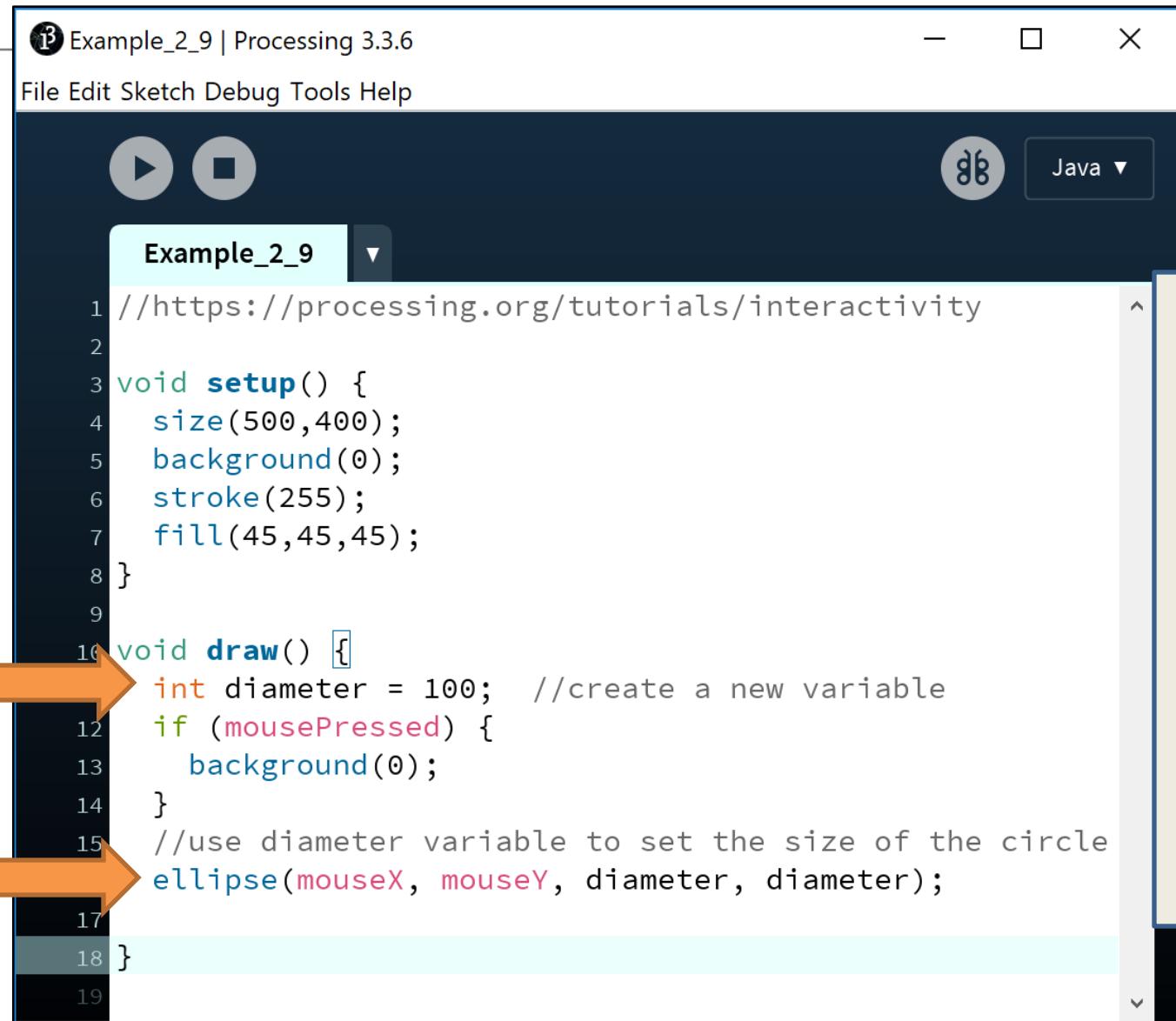
void setup() {
    size(500,400);
    background(0);
    stroke(255);
    fill(45,45,45);
}

void draw() {
    if (mousePressed) {
        background(0);
    }

    ellipse(mouseX, mouseY, 100, 100);
}
```

In this example, we have “hard coded” the value of 100 for the diameter of the circle.

Processing Example 2.9



Example_2_9 | Processing 3.3.6

File Edit Sketch Debug Tools Help

Java ▾

Example_2_9

```
1 //https://processing.org/tutorials/interactivity
2
3 void setup() {
4     size(500,400);
5     background(0);
6     stroke(255);
7     fill(45,45,45);
8 }
9
10 void draw() {
11     int diameter = 100; //create a new variable
12     if (mousePressed) {
13         background(0);
14     }
15     //use diameter variable to set the size of the circle
16     ellipse(mouseX, mouseY, diameter, diameter);
17 }
18 }
```

Here, we have replaced the “hard coded” 100 with a variable **diameter**, whose value is 100.

Local Scope – diameter variable

- The **diameter** variable is declared in the `draw()` function
i.e. it is a **local** variable.
- It is only “alive” while the `draw()` function is running.

```
void draw() {  
    int diameter = 100; //create a new variable  
    if (mousePressed) {  
        background(0);  
    }  
    //use diameter variable to set the size of the circle  
    ellipse(mouseX, mouseY, diameter, diameter);  
}
```

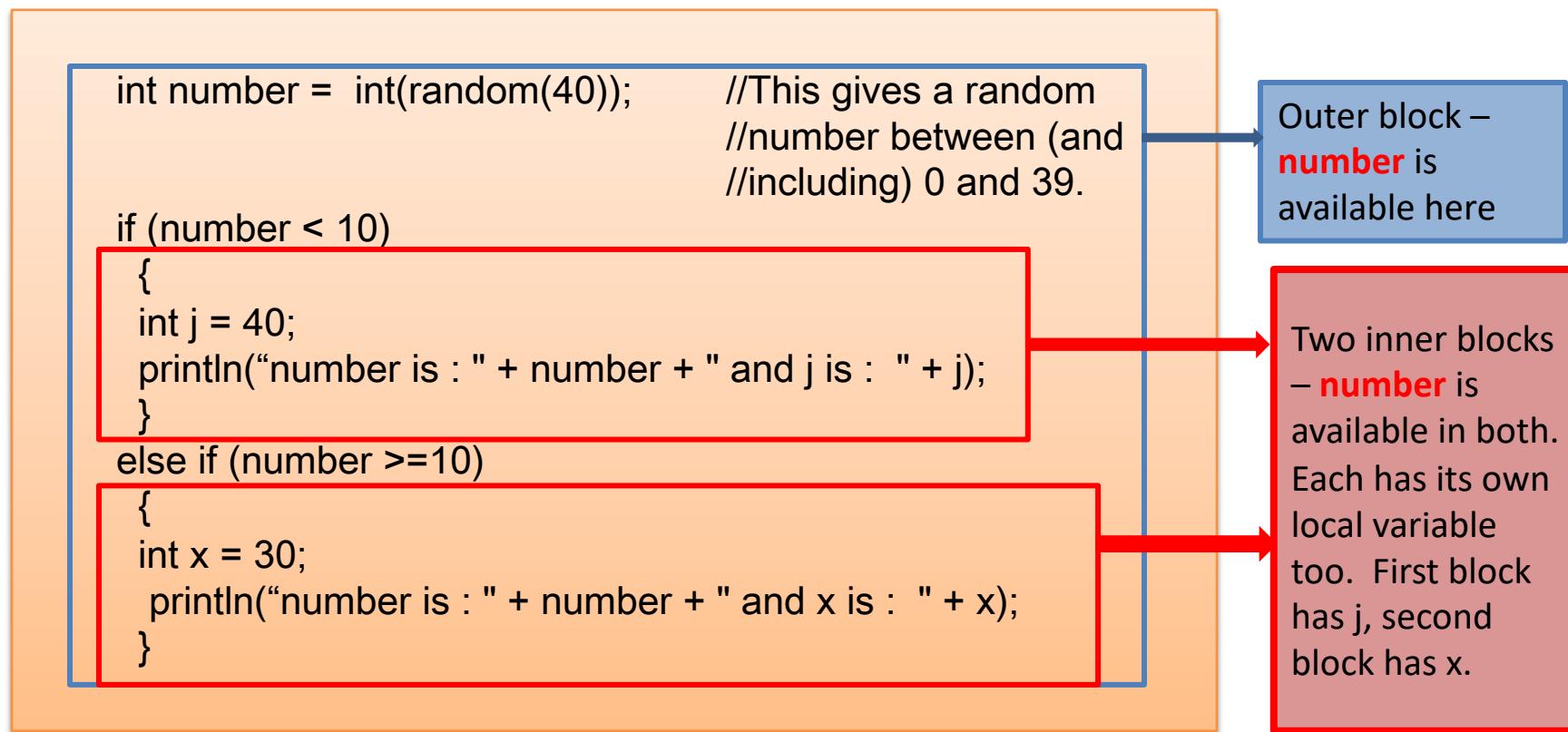
Local Scope – diameter variable

- Each time the draw() function:
 - finishes running, the **diameter** variable is destroyed.
 - is called, the **diameter** variable is re-created.

```
void draw() {  
    int diameter = 100; //create a new variable  
    if (mousePressed) {  
        background(0);  
    }  
    //use diameter variable to set the size of the circle  
    ellipse(mouseX, mouseY, diameter, diameter);  
}
```

Local variables – scope rules

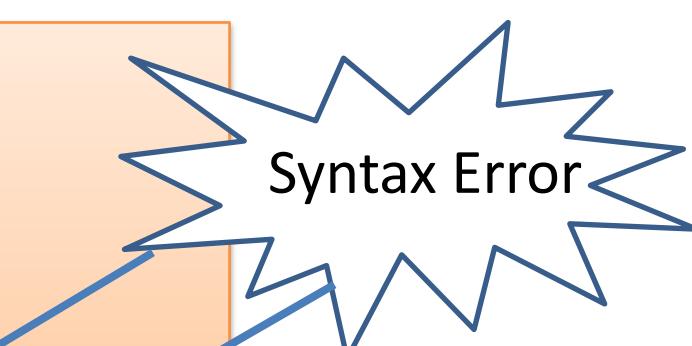
- The **scope** of a local variable is the **block** it is declared in. A block is delimited by the **curly braces {}**.
- A program can have many **nested blocks**.



Local variables – scope rules

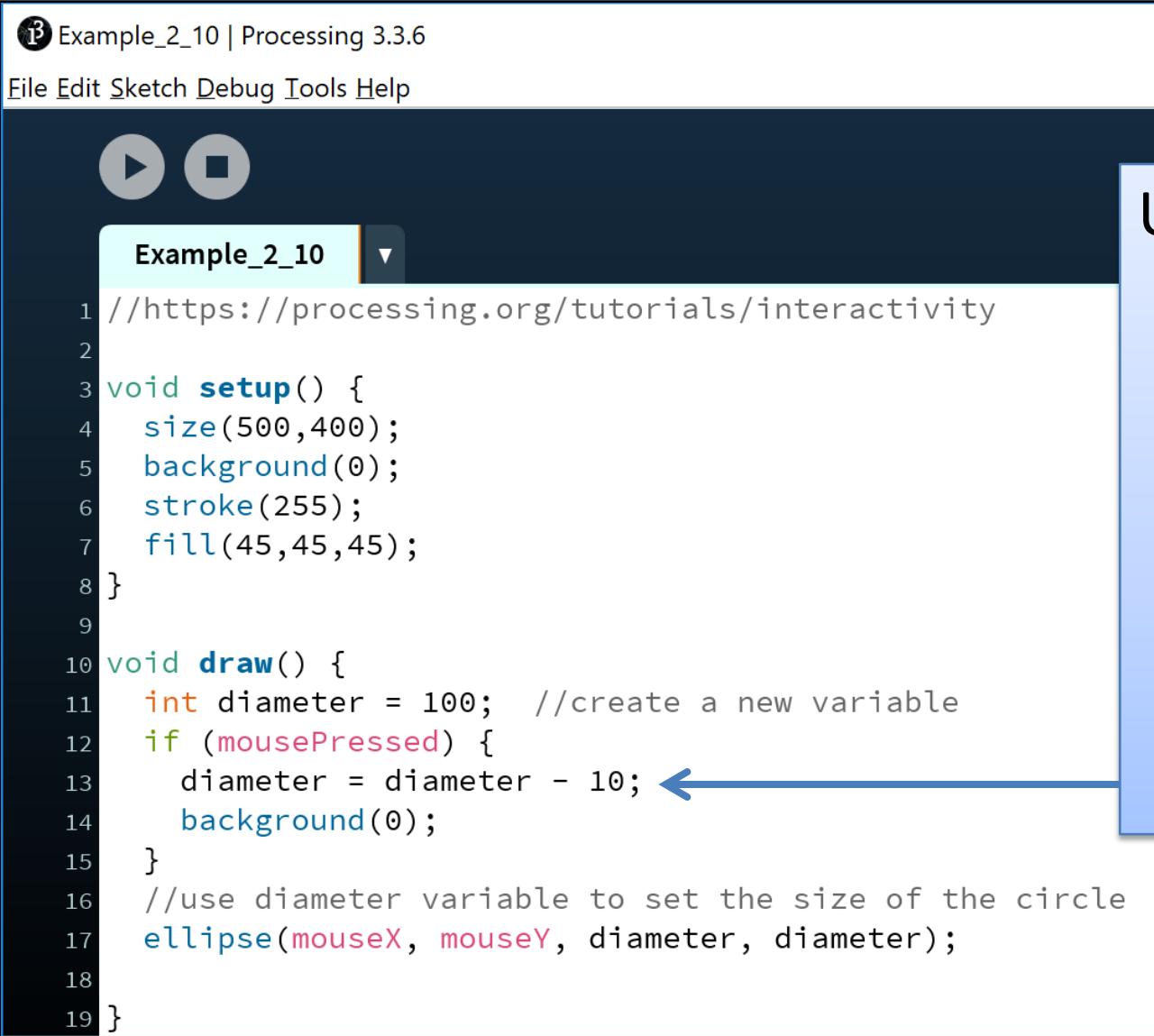
- The **lifetime** of a local variable is the time of execution of the block it is declared in.
- Trying to access a local variable outside its scope will trigger a **syntax error** e.g.:

```
void draw()
{
    if (mousePressed)
    {
        int diameter = 100;
        background(0);
    }
    ellipse(mouseX, mouseY, diameter, diameter);
}
```



Syntax Error

Processing Example 2.10



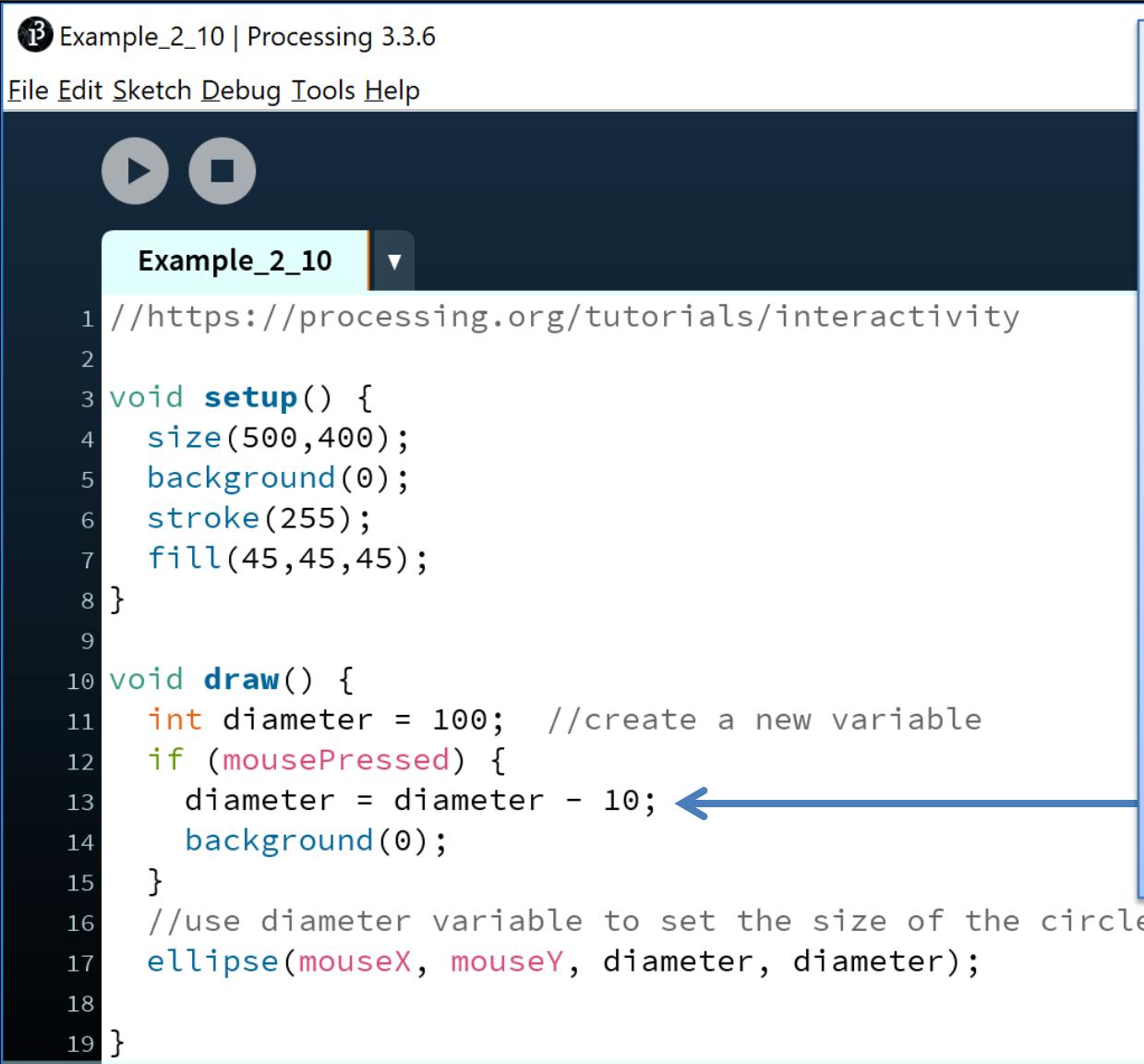
The screenshot shows the Processing 3.3.6 IDE interface. The title bar says "Example_2_10 | Processing 3.3.6". The menu bar includes "File", "Edit", "Sketch", "Debug", "Tools", and "Help". Below the menu is a toolbar with play and stop buttons. The code editor window contains the following Pseudocode:

```
1 //https://processing.org/tutorials/interactivity
2
3 void setup() {
4     size(500,400);
5     background(0);
6     stroke(255);
7     fill(45,45,45);
8 }
9
10 void draw() {
11     int diameter = 100; //create a new variable
12     if (mousePressed) {
13         diameter = diameter - 10; ←
14         background(0);
15     }
16     //use diameter variable to set the size of the circle
17     ellipse(mouseX, mouseY, diameter, diameter);
18 }
19 }
```

Using our 2.9 code, we now want to reduce the diameter size by 10 each time the mouse is pressed.

Q: Is this correct?

Processing Example 2.10



The screenshot shows the Processing IDE interface with the title bar "Example_2_10 | Processing 3.3.6" and menu bar "File Edit Sketch Debug Tools Help". Below the menu is a toolbar with play and stop buttons. The code editor contains the following Pseudocode:

```
1 //https://processing.org/tutorials/interactivity
2
3 void setup() {
4     size(500,400);
5     background(0);
6     stroke(255);
7     fill(45,45,45);
8 }
9
10 void draw() {
11     int diameter = 100; //create a new variable
12     if (mousePressed) {
13         diameter = diameter - 10; ←
14         background(0);
15     }
16     //use diameter variable to set the size of the circle
17     ellipse(mouseX, mouseY, diameter, diameter);
18 }
19 }
```

A blue callout box on the right side of the code editor highlights the line "diameter = diameter - 10;" with the text "A: We have a bug in our logic." and an explanation.

A: We have a bug in our logic.

As the **diameter** variable is re-created each time `draw()` is called, its value will be reset to 100 and will lose our previous decrement of 10. Our circle will keep resetting itself to a diameter of 100.

Global variables – scope rules!

- The **scope** of the **diameter** variable is too narrow;
 - as soon as `draw()` finishes running, the local variable is destroyed and we loose all data.
 - when `draw()` is called again, the diameter variable is recreated and its value is set to 100.
- We need a **diameter** variable that lives for the **lifetime** of a sketch i.e.
 - a global variable.

Processing Example 2.11

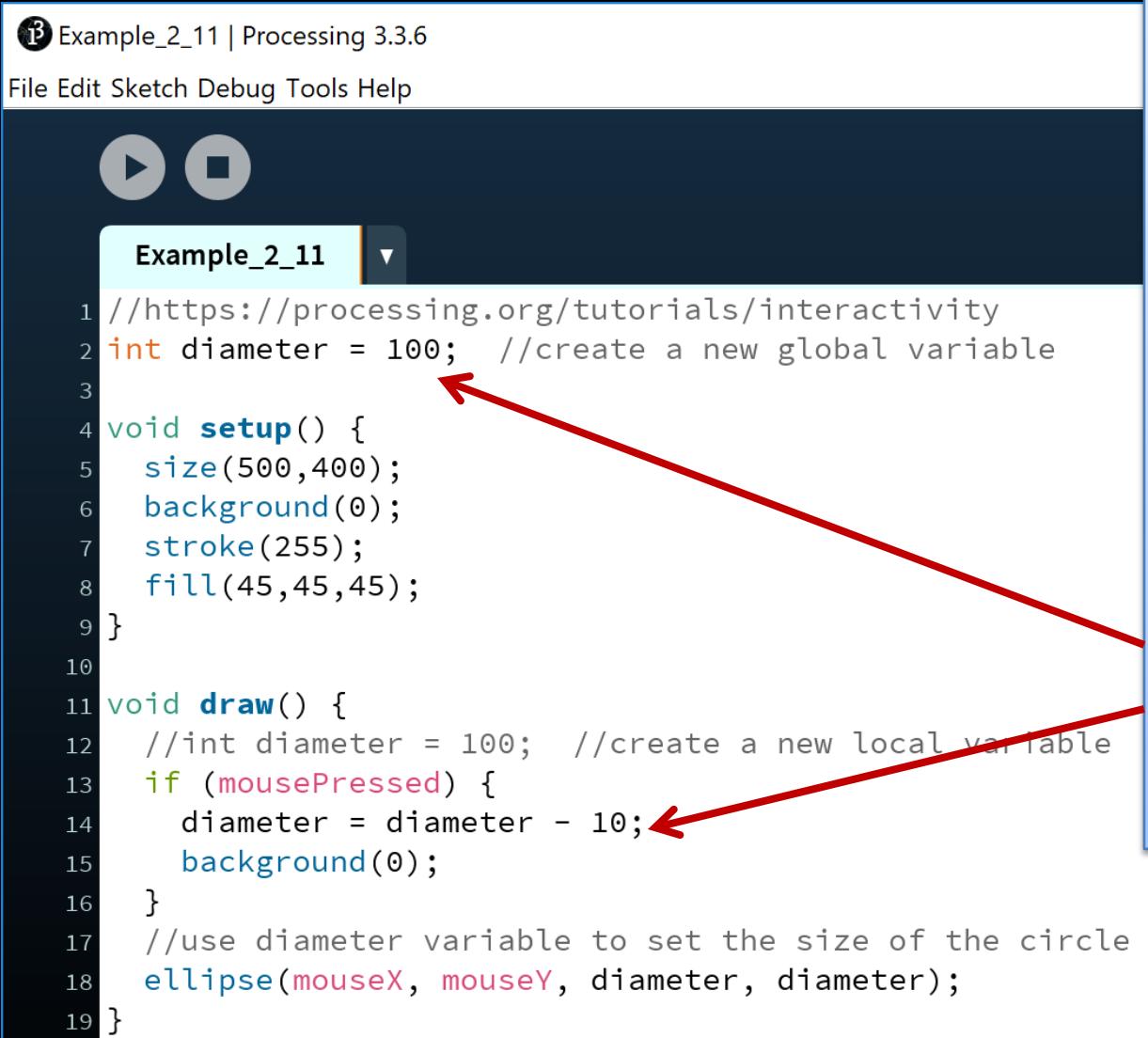
```
//https://processing.org/tutorials/interactivity
int diameter = 100; //create a new global variable
void setup() {
    size(500,400);
    background(0);
    stroke(255);
    fill(45,45,45);
}
void draw() {
    //int diameter = 100; //create a new local variable
    if (mousePressed) {
        diameter = diameter - 10;
        background(0);
    }
    //use diameter variable to set the size of the circle
    ellipse(mouseX, mouseY, diameter, diameter);
}
```

Let's try fix the bug!

We established that the **scope** of the **local diameter** variable was too narrow; **diameter** is recreated each time **draw()** is called and its value is set to 100.

Comment out the local **diameter** variable and instead make it **global scope**.

Processing Example 2.11



```
//https://processing.org/tutorials/interactivity
int diameter = 100; //create a new global variable
void setup() {
    size(500,400);
    background(0);
    stroke(255);
    fill(45,45,45);
}
void draw() {
    //int diameter = 100; //create a new local variable
    if (mousePressed) {
        diameter = diameter - 10;
        background(0);
    }
    //use diameter variable to set the size of the circle
    ellipse(mouseX, mouseY, diameter, diameter);
}
```

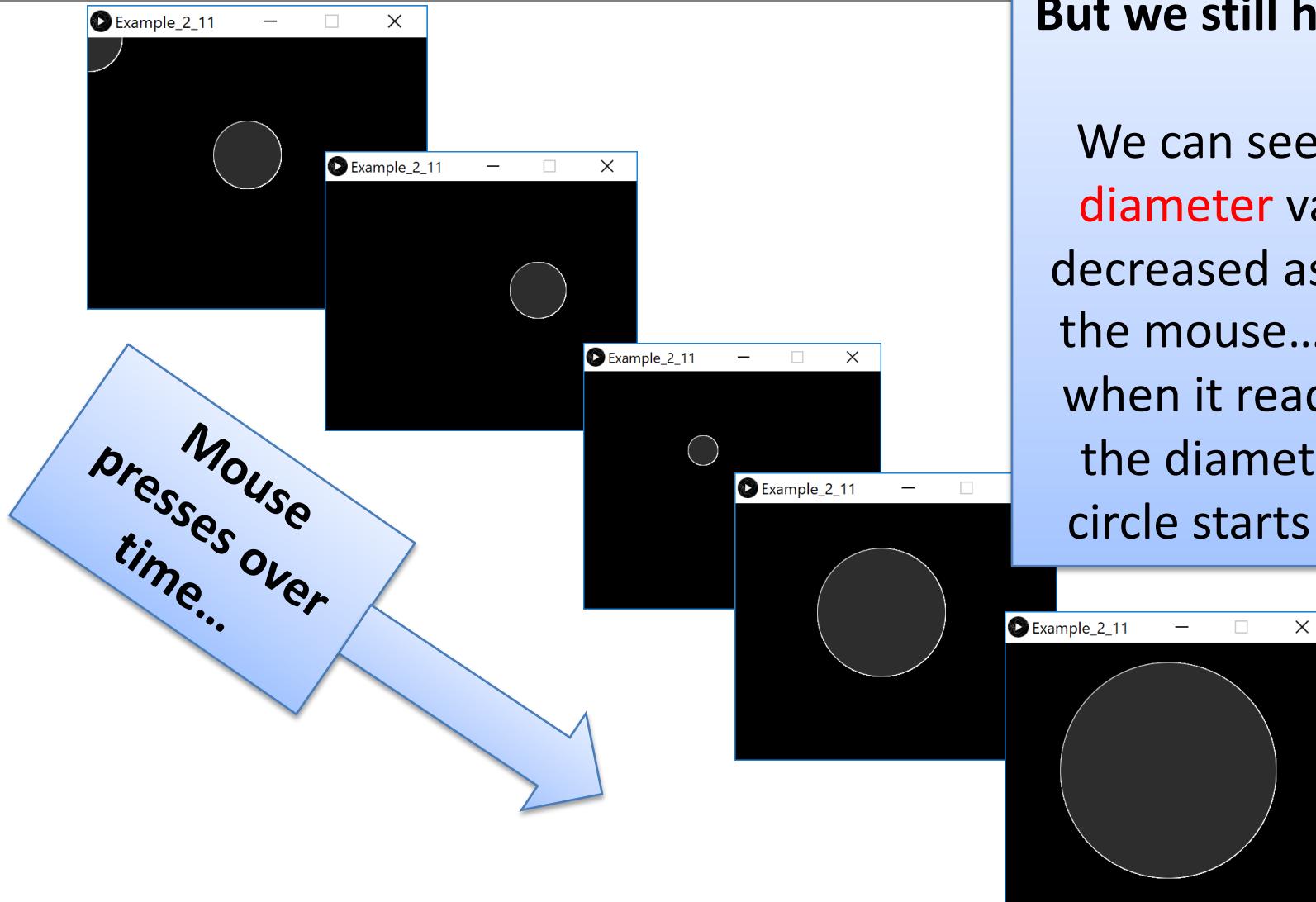
But we still have a bug!

The **diameter** variable is decreased each time we press the mouse.

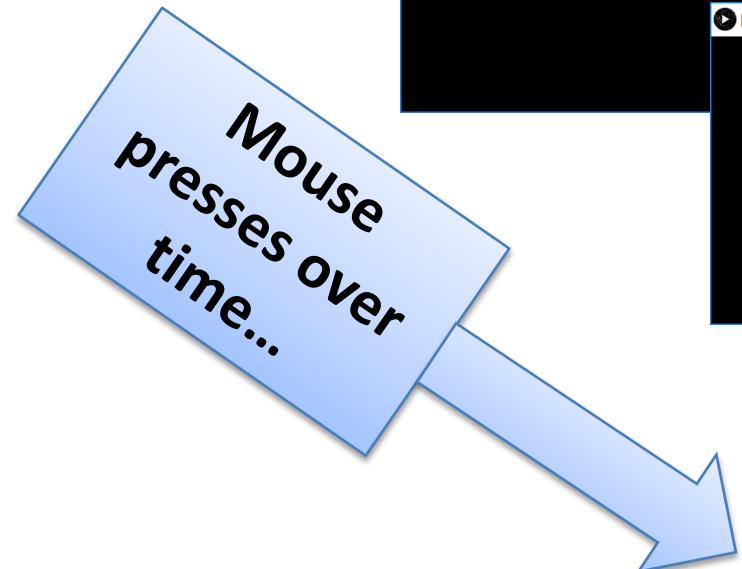
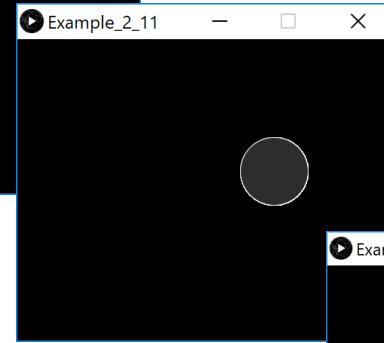
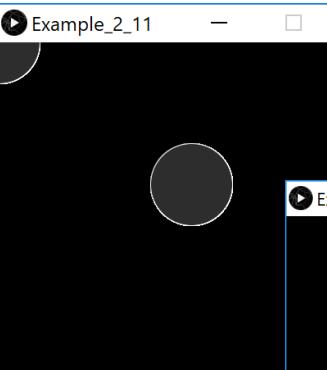
Correct!

Q: However, what happens when the mouse pressing reduces the value of **diameter** to zero?

Processing Example 2.11

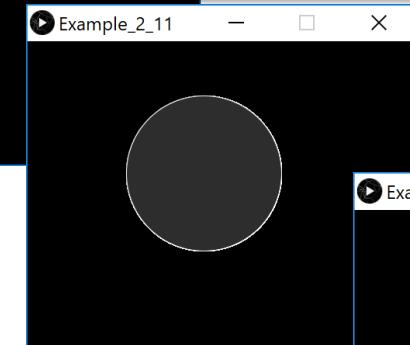


Processing Example 2.11

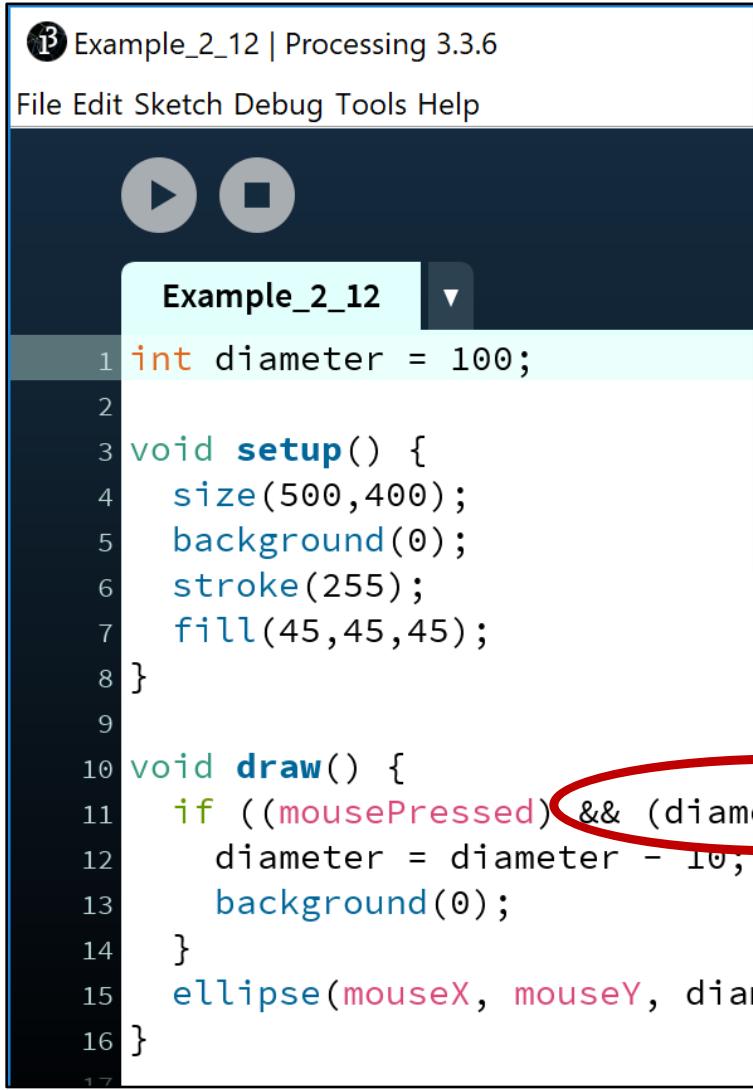


What is happening?

The **width** and **height** in the **ellipse** function are **absolute values** (negative sign is dropped). So, even though **diameter** had a value of say, -50, the **magnitude** is all that is used when drawing the ellipse...i.e. 50.



Processing Example 2.12

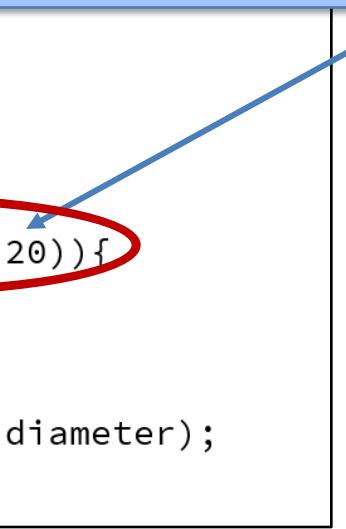


The screenshot shows the Processing 3.3.6 IDE interface with the title bar "Example_2_12 | Processing 3.3.6" and menu bar "File Edit Sketch Debug Tools Help". Below the menu is a toolbar with play and stop buttons. The sketch window displays the code for "Example_2_12". The code defines a variable "diameter" and sets up a window of size 500x400 with a background color of 0. It also sets stroke and fill colors to 255, 45, 45. In the draw() function, it checks if the mouse is pressed and if the diameter is greater than 20. If both conditions are true, it reduces the diameter by 10 and draws an ellipse at the mouse position with the current diameter value.

```
1 int diameter = 100;
2
3 void setup() {
4     size(500,400);
5     background(0);
6     stroke(255);
7     fill(45,45,45);
8 }
9
10 void draw() {
11     if ((mousePressed) && (diameter > 20)){
12         diameter = diameter - 10;
13         background(0);
14     }
15     ellipse(mouseX, mouseY, diameter, diameter);
16 }
```

In the **ellipse** function, the width and height are absolute values (negative sign is dropped).

To handle this logic bug, we need to stop reducing the **diameter** by 10 when we reach a certain value, say 20.



Processing Example 2.12

```
Example_2_12 | Processing 3.3.6
File Edit Sketch Debug Tools Help

Example_2_12

1 int diameter = 100;
2
3 void setup() {
4     size(500,400);
5     background(0);
6     stroke(255);
7     fill(15,15,45),
8     frameRate(20); //slow down the frame refresh,
9                 //from default 60 to 20 per second
10}
11
12 void draw() {
13     if ((mousePressed) && (diameter > 20)){
14         diameter = diameter - 10;
15         background(0);
16     }
17     ellipse(mouseX, mouseY, diameter, diameter);
18}
```

When you run this code, it appears the reduction is larger than 10 when we press the mouse?

Why? The default frame rate is 60 refreshes of the screen per second i.e. draw() is called 60 times per second.

You can change the frame rate by calling the **frameRate()** function.

Topics list

1. Use of **println()**, **text()** in Processing
2. Variable **Scope**
3. Compound Assignment Statements

Compound Assignment Statements

	Full statement	Shortcut
Mathematical shortcuts	$x = x + a;$	$x += a;$
	$x = x - a;$	$x -= a;$
	$x = x * a;$	$x *= a;$
	$x = x/a;$	$x /= a;$
Increment shortcut	$x = x+1;$	$x++;$
Decrement shortcut	$x = x - 1;$	$x--;$

Questions?

