Programming Fundamentals 1

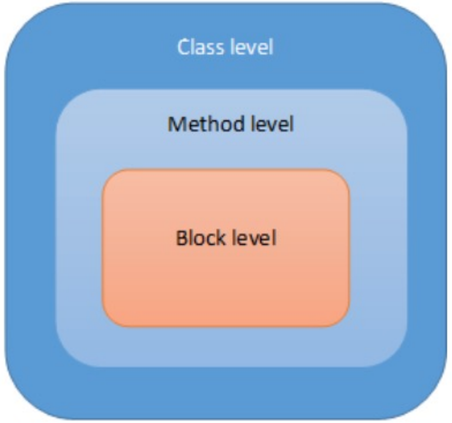
Produced by
Mr. Dave Drohan (david.drohan@setu.ie)
Dr. Siobhán Drohan
Ms. Mairead Meagher

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

Introduction to Processing

Scope of variables, Printing and Compound Assignment Statements

Scope of Variables



scope · compound assignments

Agenda

- ❑ Use of println(), text() in Processing

- ❑ Variable Scope

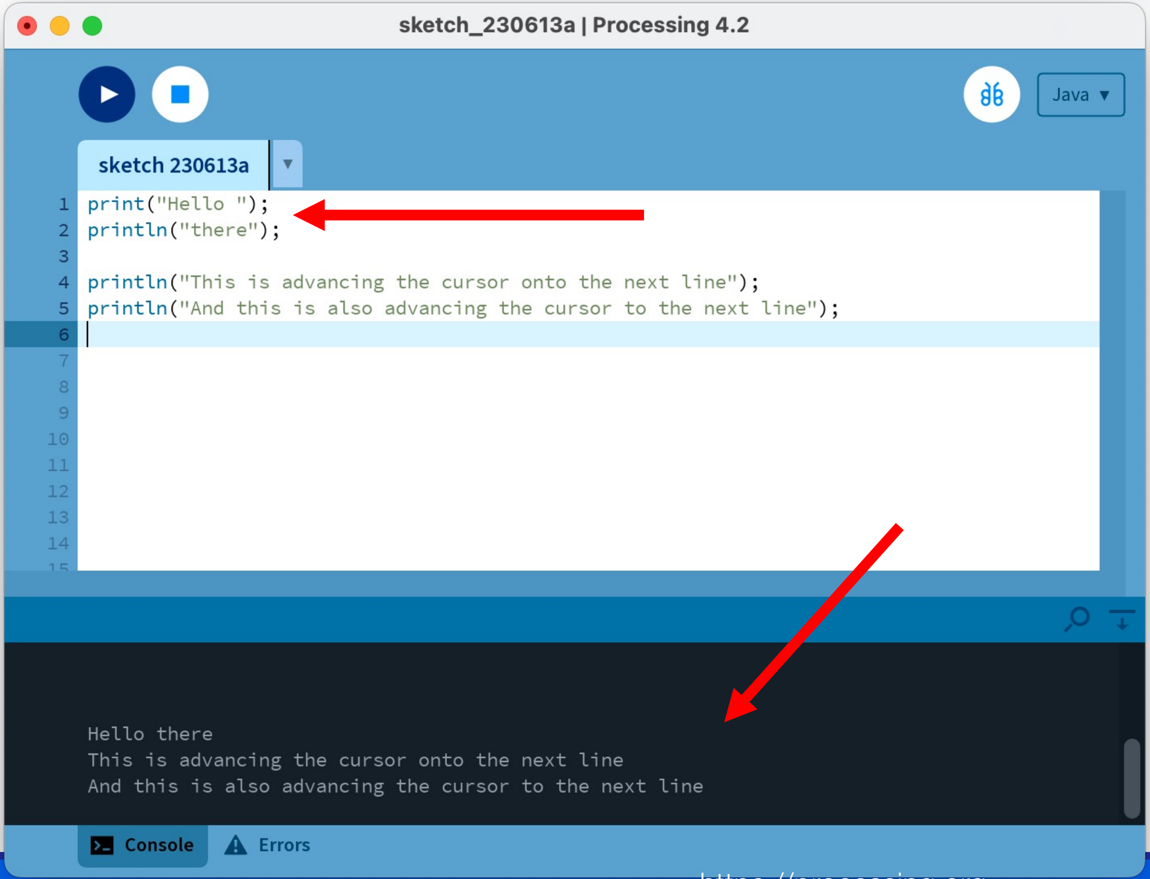
- ❑ Compound Assignment Statements

Use of println(), text() in Processing

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()

println() and text() in Processing

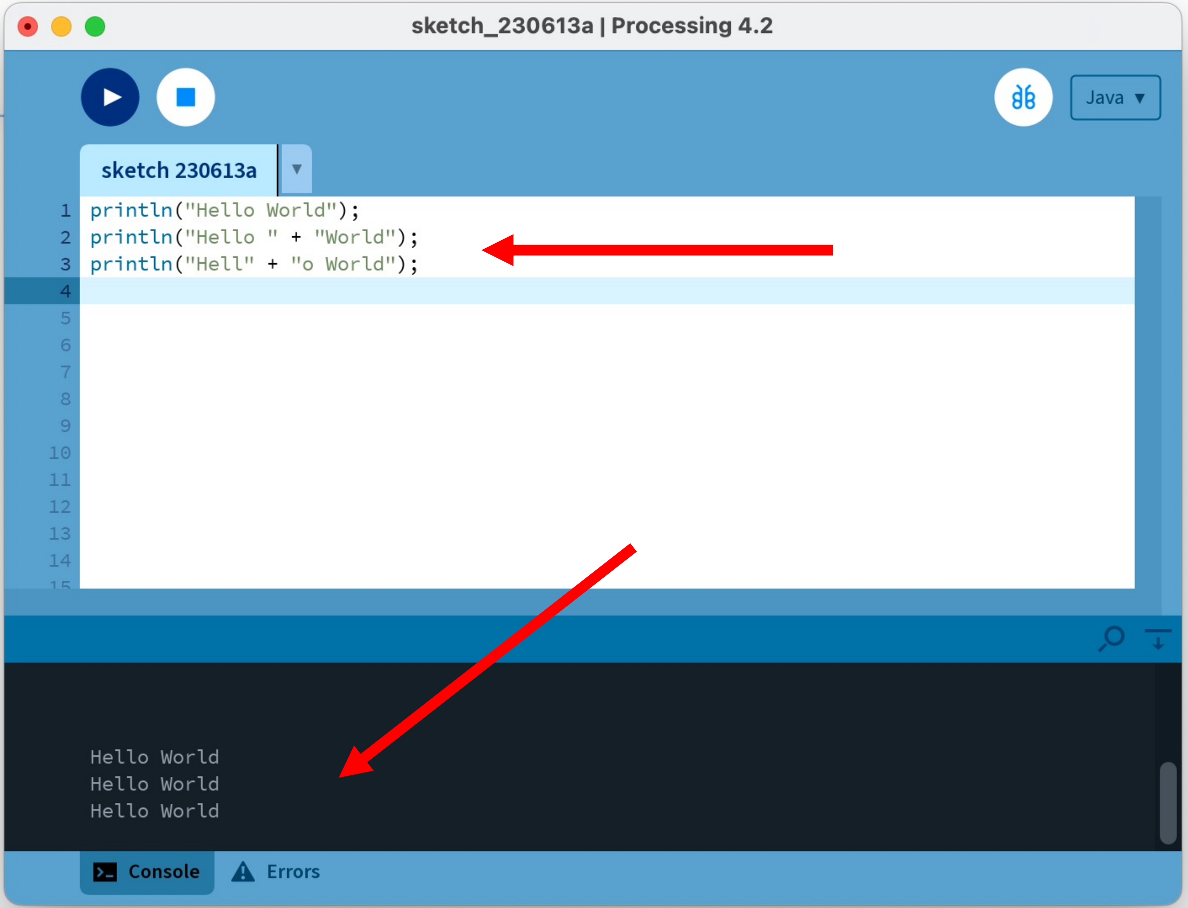


```
print("Hello ");
println("there");

println("This is advancing the cursor onto the next line");
println("And this is also advancing the cursor to the next line");
```

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

println()

Each statement prints the same output.

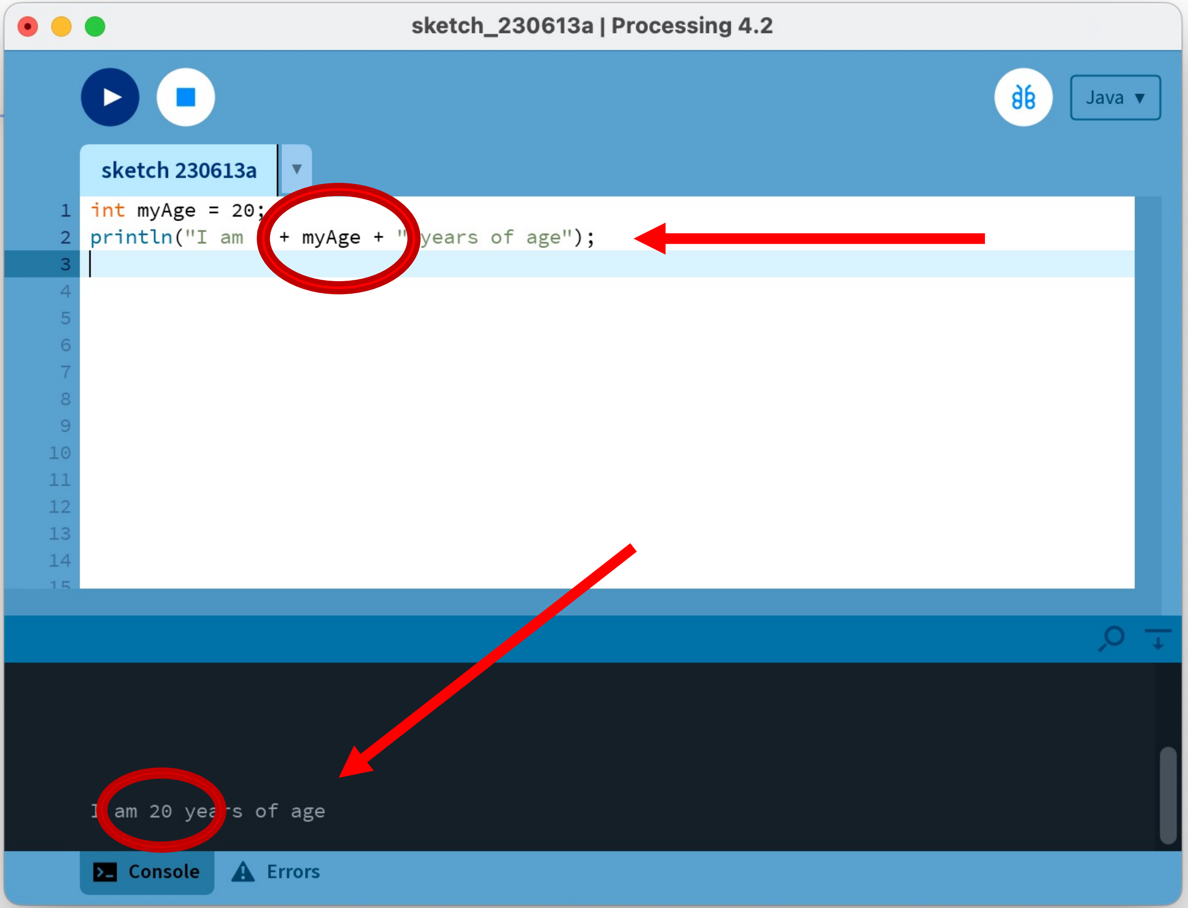


```
println("Hello World");
println("Hello " + "World");
println("Hell" + "o World");
```

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

println()

We can use variables in the print statement.



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

```
I am 20 years of age
```

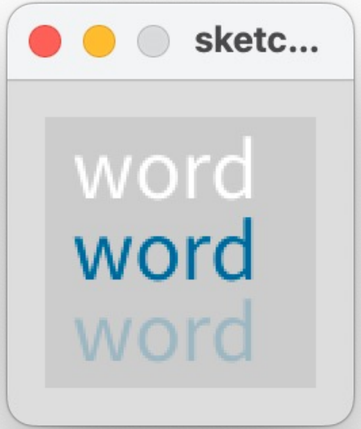
text() in processing

- 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

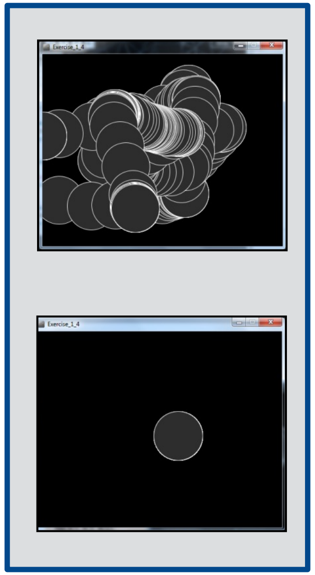


Variable Scope

Recap: Processing Example 3.8

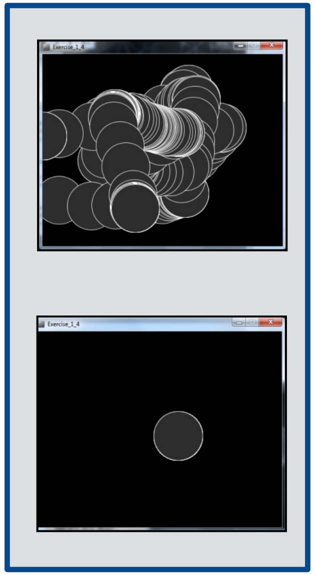
Functionality:

- 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.

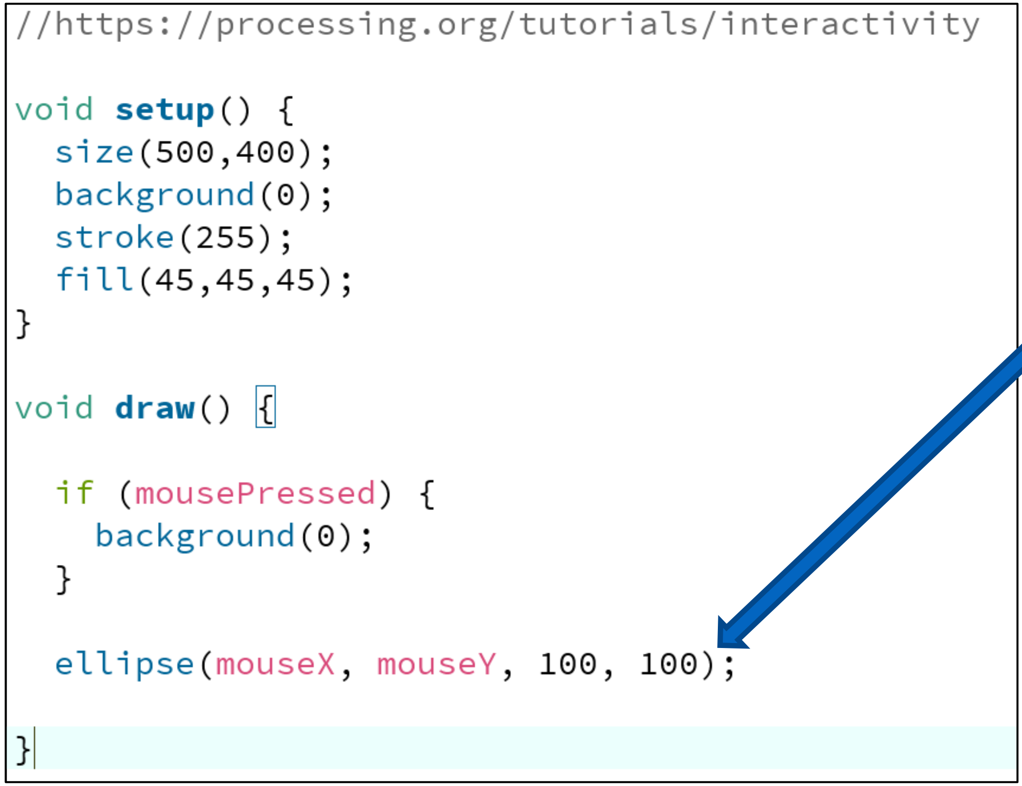


Recap: Processing Example 3.8

```
//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);

}
```



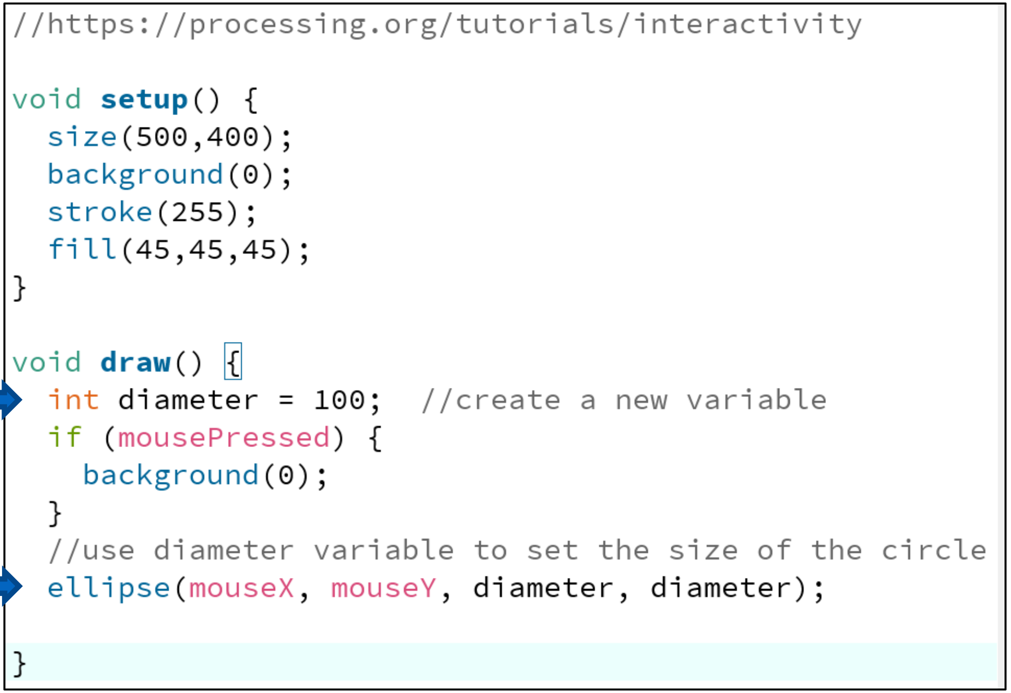
Recap: Processing Example 3.8



```
//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 4.1



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

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

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);

}
```

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);

}
```

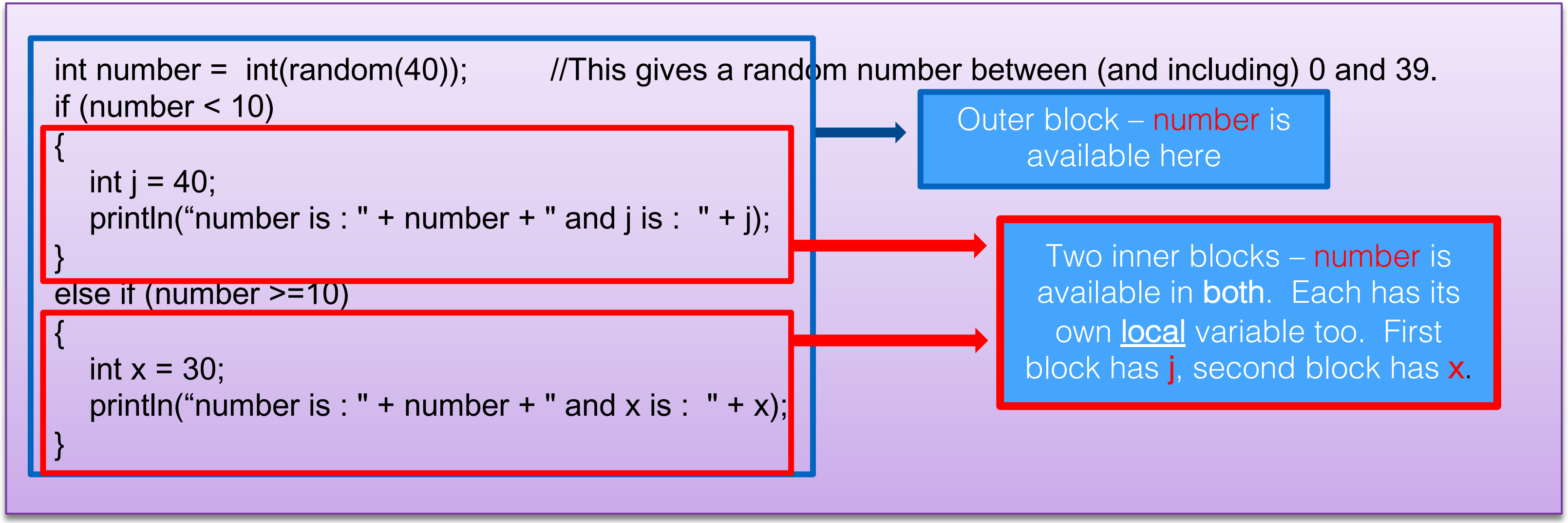
Local Scope – diameter variable

- 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.**



```
int number =  int(random(40));        //This gives a random number between (and including) 0 and 39.
if (number < 10)
{
    int j = 40;
    println(“number is : " + number + " and j is :  " + j);
}
else if (number >=10)
{
    int x = 30;
    println(“number is : " + number + " and x is :  " + x);
}
```

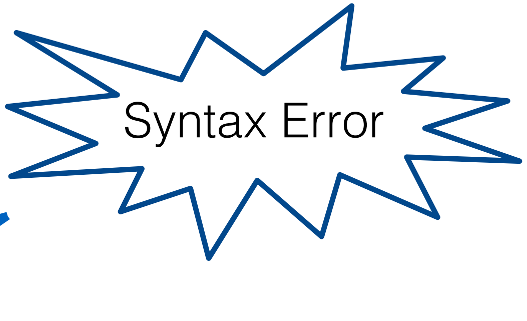
Outer block – number is available here

Two inner blocks – number is available in **both**. Each has its own <u>local</u> variable too. First block has j, second block has x.

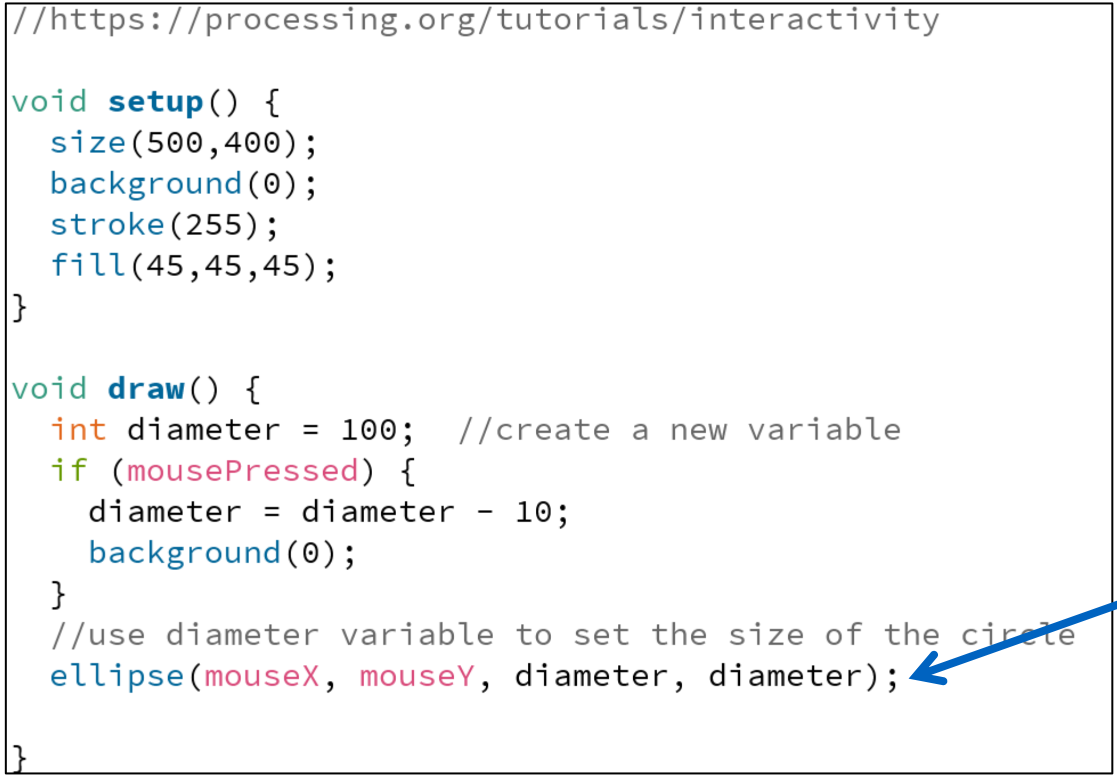
Local variables – scope rules

- 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);
}
```



Processing Example 4.2



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

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

void draw() {
  int diameter = 100;  //create a new variable
  if (mousePressed) {
    diameter = diameter - 10;
    background(0);
  }
  //use diameter variable to set the size of the circle
  ellipse(mouseX, mouseY, diameter, diameter);

}
```

Using our 4.1 code, we now want to reduce the diameter size by 10 each time the mouse is pressed.

Q: Is this correct?

Processing Example 4.2

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

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

void draw() {
  int diameter = 100;   //create a new variable
  if (mousePressed) {
    diameter = diameter - 10;
    background(0);
  }
  //use diameter variable to set the size of the circle
  ellipse(mouseX, mouseY, diameter, diameter);

}
```

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 4.3

```
//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 4.3

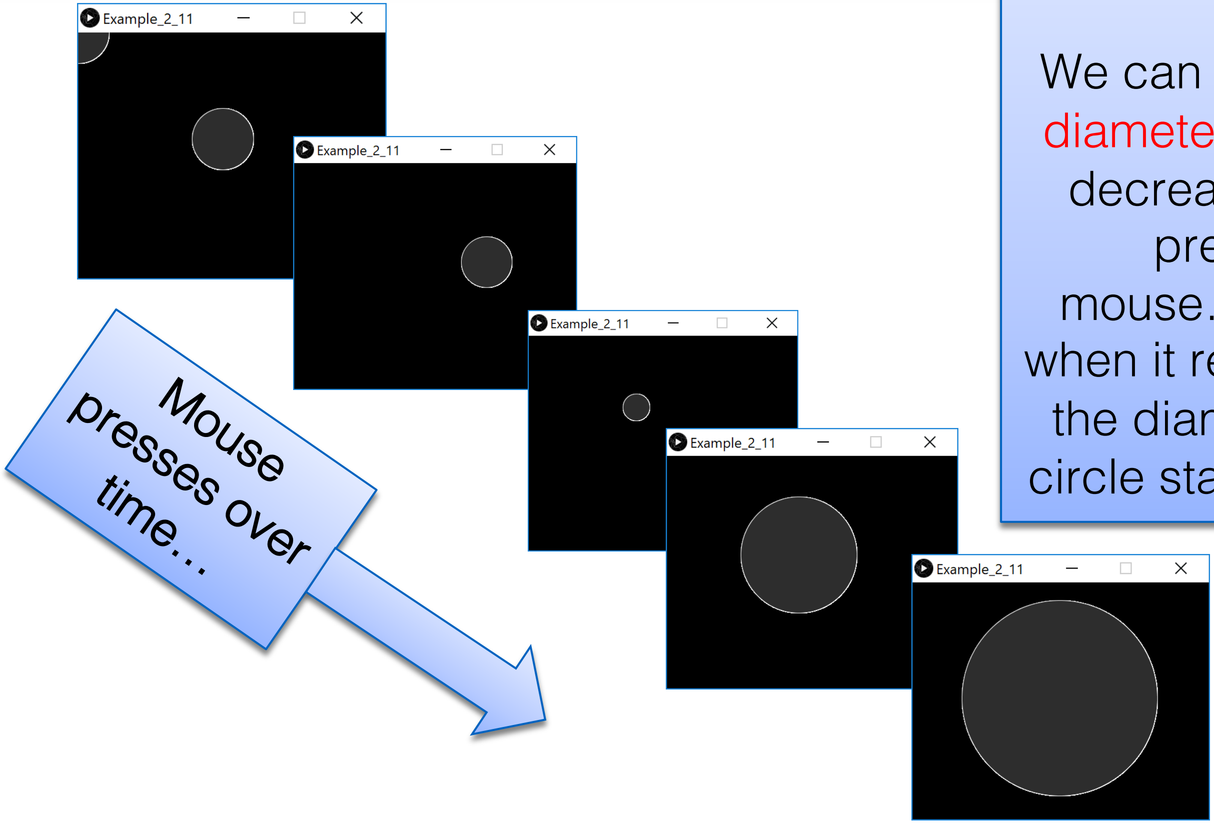
```
//https://processing.org/tutorials/interactivity
int diameter = 100;  //create a new global variable

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 4.3

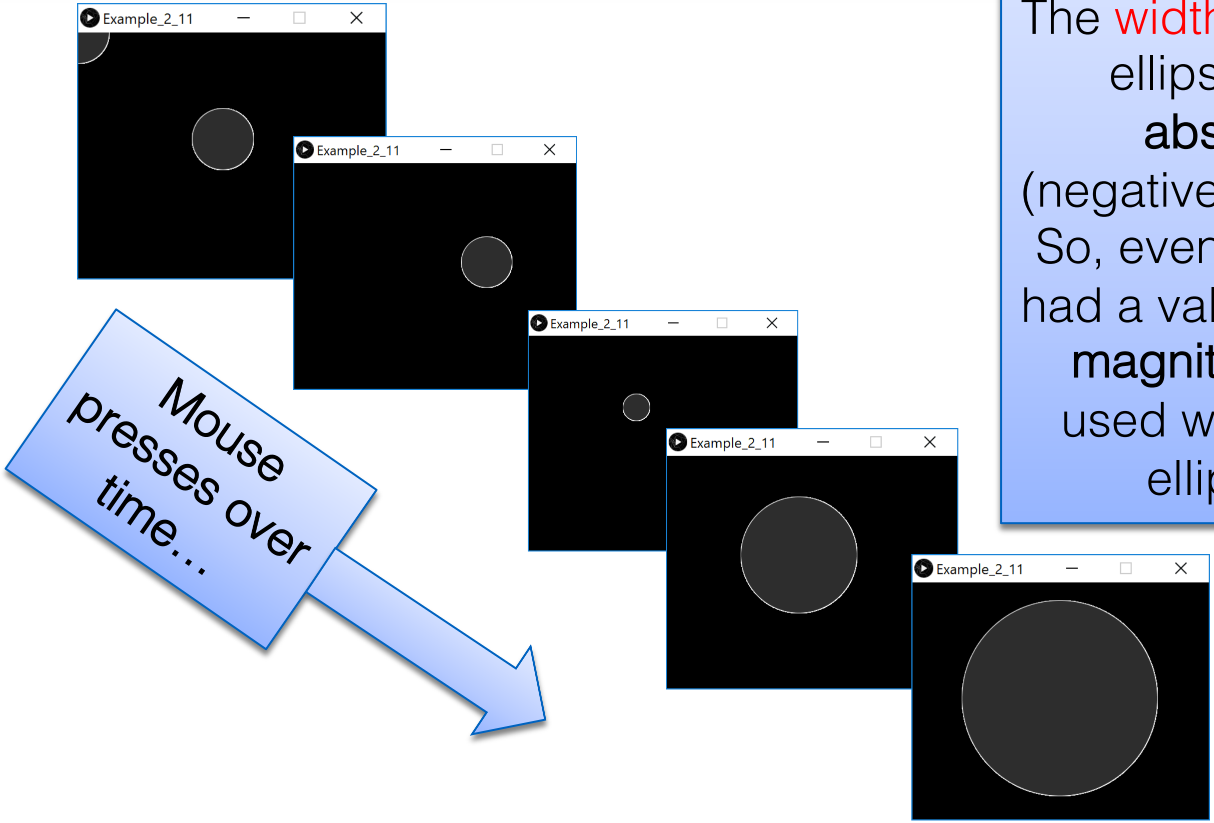


Mouse presses over time…

But we still have a bug

We can see that the diameter variable is decreased as we press the mouse…however, when it reaches zero, the diameter of the circle starts growing!

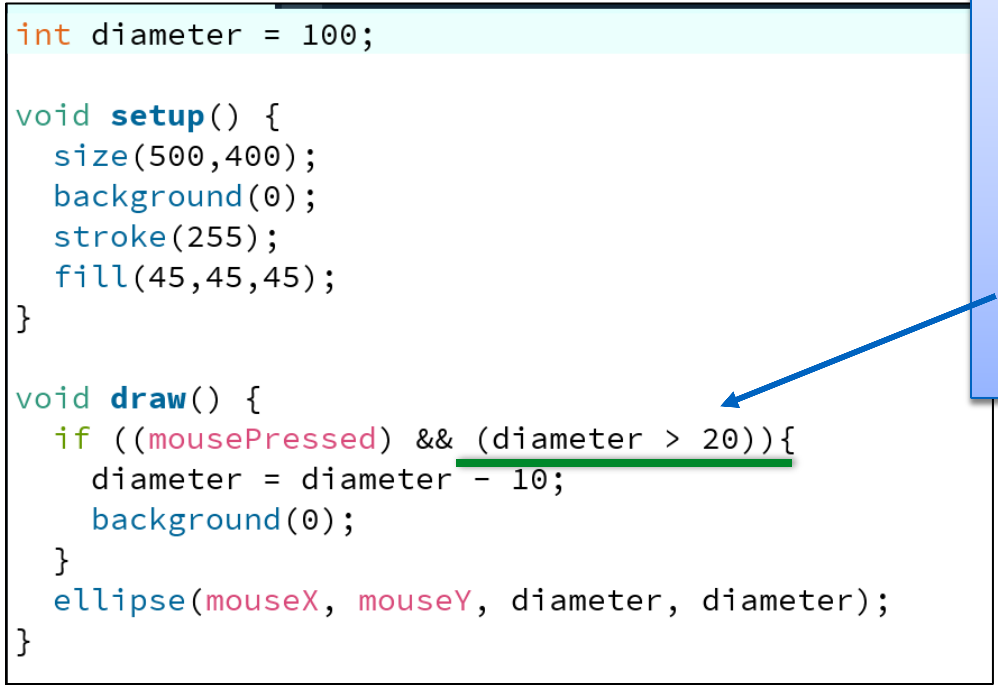
Processing Example 4.3



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 4.4

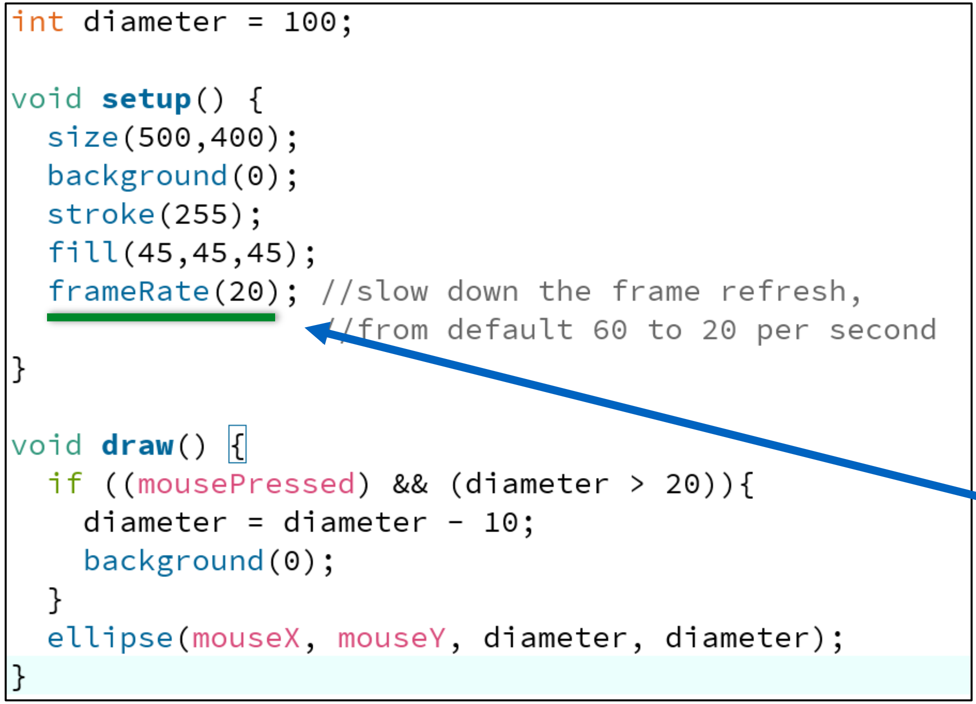


```
int diameter = 100;

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

void draw() {
  if ((mousePressed) && (diameter > 20)){
    diameter = diameter - 10;
    background(0);
  }
  ellipse(mouseX, mouseY, diameter, diameter);
}
```

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 4.4



```
int diameter = 100;

void setup() {
  size(500,400);
  background(0);
  stroke(255);
  fill(45,45,45);
  frameRate(20); //slow down the frame refresh,
                 //from default 60 to 20 per second
}

void draw() {
  if ((mousePressed) && (diameter > 20)){
    diameter = diameter - 10;
    background(0);
  }
  ellipse(mouseX, mouseY, diameter, diameter);
}
```

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.

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**--**;

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?

References

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

Thanks.