# Iteration

For Loops, variable scope, compound statements

---

Produced by: Department of Computing and Mathematics

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

## Topics list

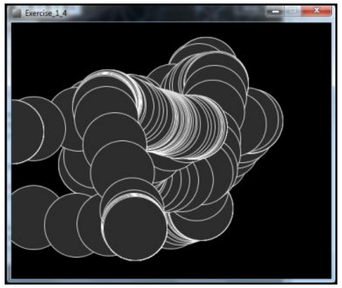
- Variable Scope
- Repetition in Programming (for loops).
- Compound Assignment Statements

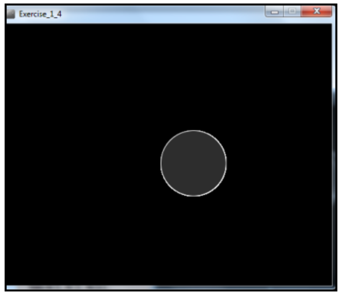
# Recap: Processing Example 5.4

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

```
void draw() {

  if (mousePressed) {
    background(0);
  }
  ellipse(mouseX, mouseY, 100, 100);
}
```



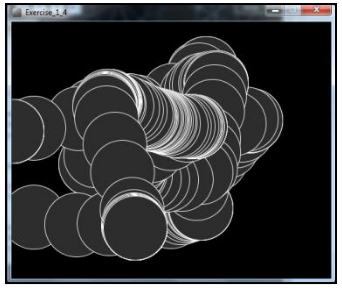


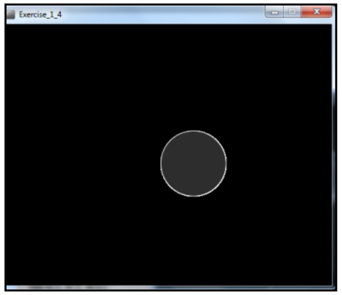
https://processing.org/tutorials/interactivity/

# Processing Example 6.1

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

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





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

# Local Scope – diameter variable

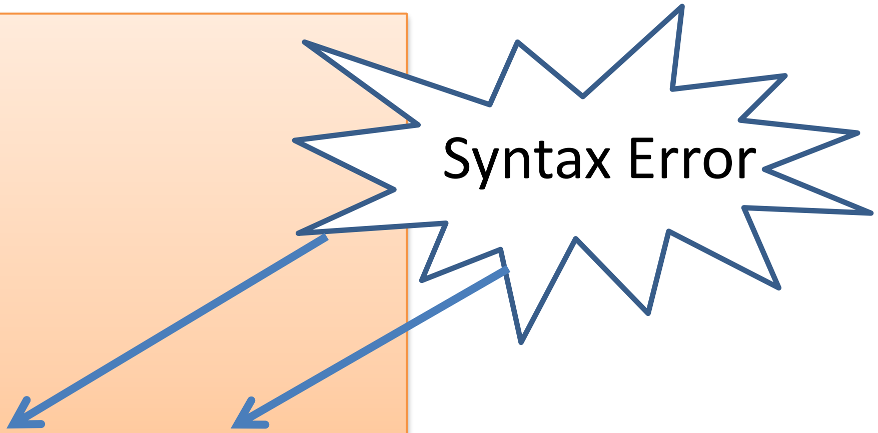
- The diameter variable is declared in the draw() function i.e. it is a local variable.
- It is only "alive" while the draw() function is running.
- Each time the draw() function:
  - finishes running, the diameter variable is destroyed.
  - is called, a new diameter variable is re-created.

```
void draw() {
  int diameter = 100;
  if (mousePressed) {
    background(0);
  }
  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 {}.
- 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);
}
```

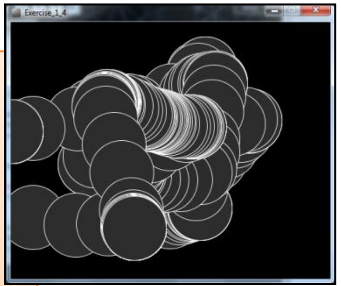


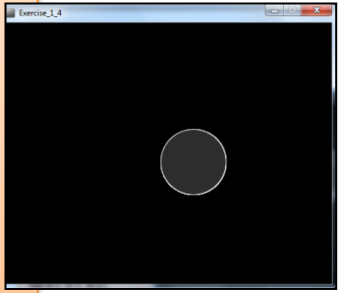
# Processing Example 6.2

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

We now want to reduce the diameter size by 10 each time the mouse is pressed.  Is this correct?

```
void draw() {
  int diameter = 100;
  if (mousePressed) {
    diameter = diameter – 10;
    background(0);
  }
  ellipse(mouseX, mouseY, diameter, diameter);
}
```





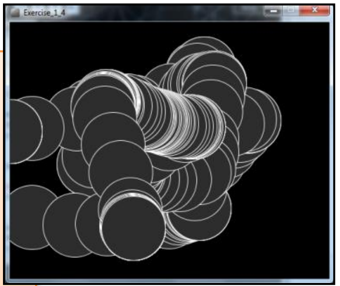
https://processing.org/tutorials/interactivity/

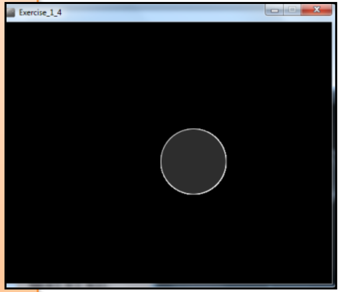
# Processing Example 6.2

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

**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 loose our decrement of 10.

```
void draw() {
  int diameter = 100;
  if (mousePressed) {
    diameter = diameter – 10;
    background(0);
  }
  ellipse(mouseX, mouseY, diameter, diameter);
}
```





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

## 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.

- We need a diameter variable that lives for the lifetime is sketch i.e. a global variable.

# Processing Example 6.3

```
int diameter = 100;

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

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

**We still have a bug in our logic.**

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

However, what happens when we reach zero?

# Processing Example 6.3

```
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 by 10 when we reach a certain value, say 20.

## Processing Example 6.3

```
int diameter = 100;

void setup() {
  size(500,400);
  background(0);
  stroke(255);
  fill(45,45,45);
  frameRate(20);
}

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

**Did you notice that it seems 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

- Variable Scope
- Repetition in Programming (for loops).
- Compound Assignment Statements

## Recap: Boolean conditions

- A boolean condition is an expression that evaluates to either true or false e.g.

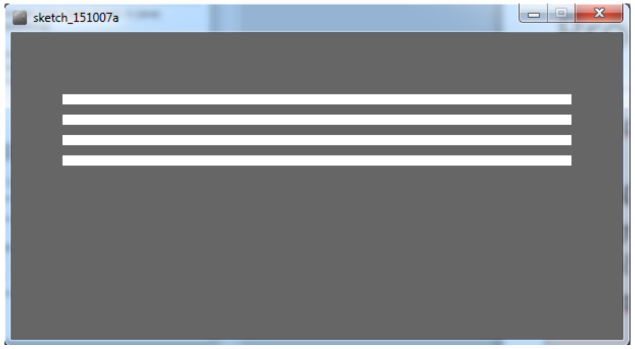
```
mouseX < 50
```

- Boolean conditions can be used to control:
  - Selection i.e. if statements and
  - Iteration i.e. loops (we will look at these now).

## Repetition in Programming

- Computers are very good at repetition.
- Draw a rectangle 4 times that has a gap of 10 pixels between each one:
  - Without loop:

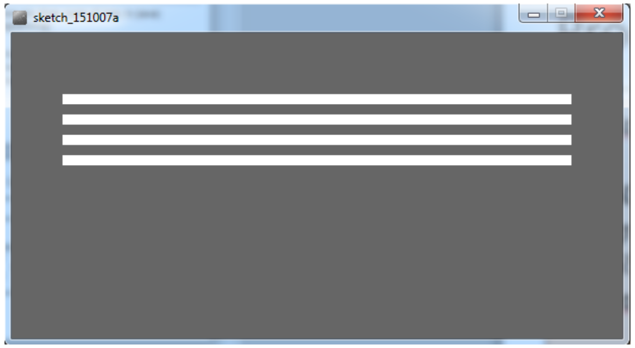
```
rect(50, 60, 500, 10);
rect(50, 80, 500, 10);
rect(50, 100, 500, 10);
rect(50, 120, 500, 10);
```



## Repetition in Programming

- Draw a rectangle 4 times that has a gap of 10 pixels between each one:
  - With a loop:
    - do this 4 times (adding 20 onto the yCoordinate variable each time).

```
rect(50, yCoordinate, 500, 10);
```



## Loops in Programming

- There are three types of loop in (Java) programming:
  - For loops (we will cover these).
  - While loops
  - Do While loops

## For loop pseudo-code

General form of a for loop

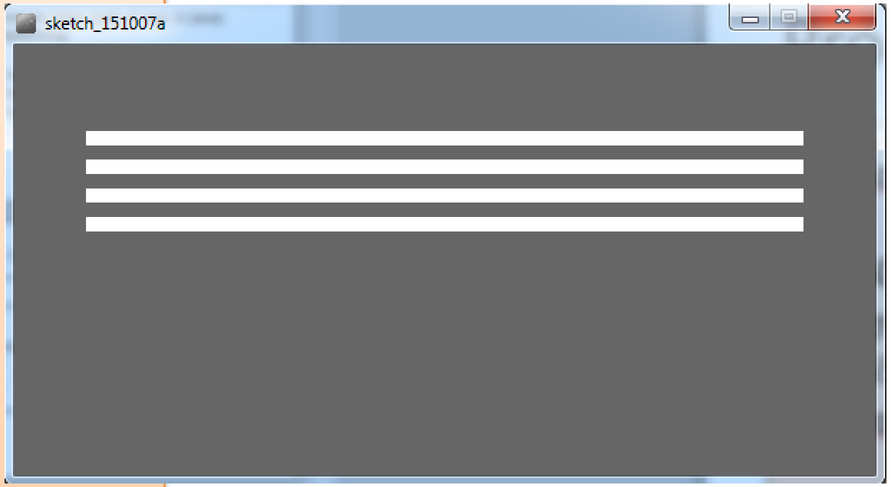
```
for(initialization; boolean condition; post-body action)
{
    statements to be repeated
}
```

## Processing Example 6.4

```
int yCoordinate = 60;

size(600, 300);
background(102);
fill(255);
noStroke();

for(int i = 0; i < 4; i++)
{
   rect(50, yCoordinate, 500, 10);
   yCoordinate = yCoordinate + 20;
}
```



# For loop syntax

```
for(int i = 0;  i < 4;  i++)
```

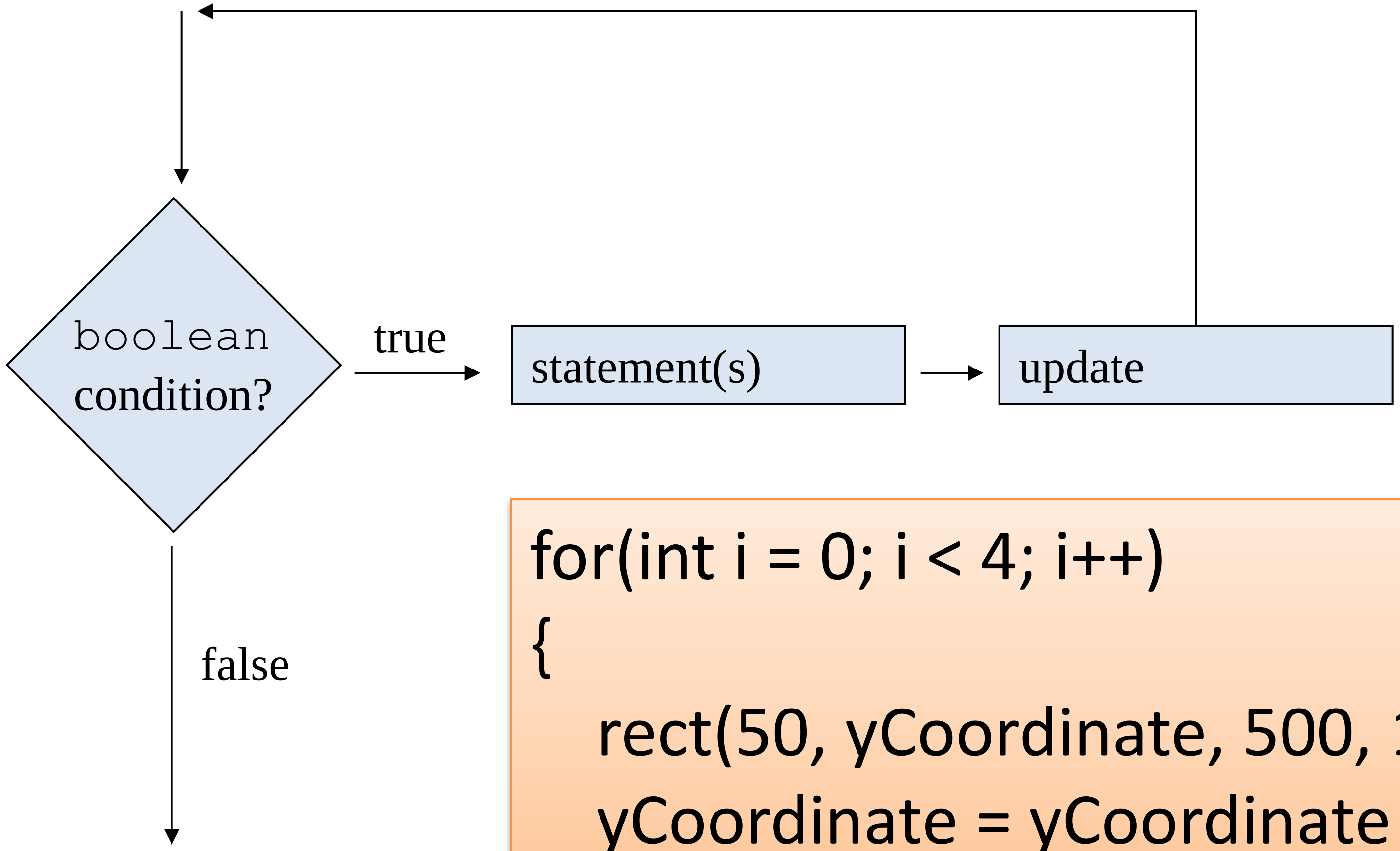
for(*initialization*; *boolean condition*; *post-body action*)
{
    *statements to be repeated*
}

## For loop syntax

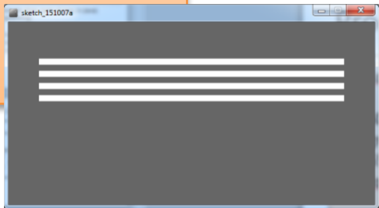
```
for(int i = 0;  i < 4;  i++)
```

| | | |
|---|---|---|
| Initialization | int i = 0 | Initialise a loop control variable (LCV) e.g. i. It can include a variable declaration. |
| Boolean condition | i < 4 | Is a valid boolean condition that typically tests the loop control variable (LCV). |
| Post-body action | i++ | A change to the loop control variable (LCV). Contains an assignment statement. |

## `for` Loop Flowchart



```
for(int i = 0; i < 4; i++)
{
    rect(50, yCoordinate, 500, 10);
    yCoordinate = yCoordinate + 20;
}
```



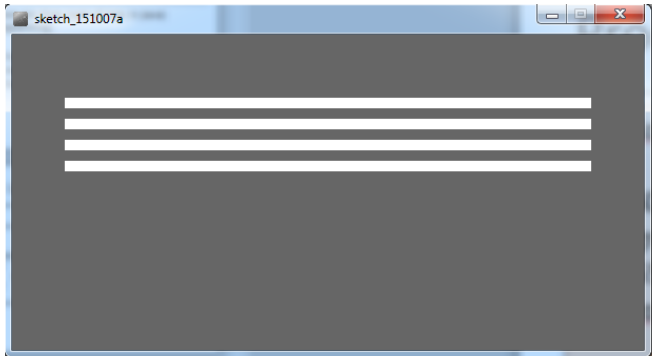
# Returning to: Processing Example 6.4

```
int yCoordinate = 60;

size(600, 300);
background(102);
fill(255);
noStroke();

for(int i = 0; i < 4; i++)
{
   rect(50, yCoordinate, 500, 10);
   yCoordinate = yCoordinate + 20;
}
```

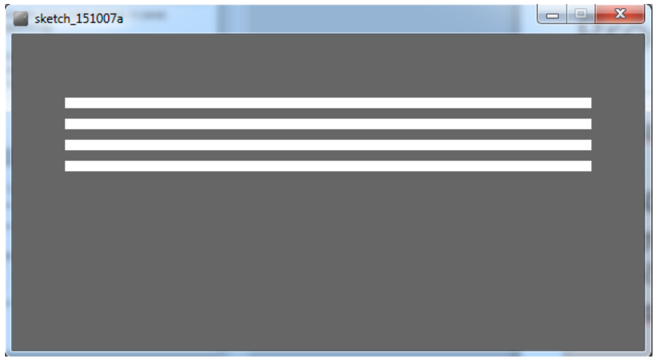
Do we need the yCoordinate variable? Can you think of a different approach using a for loop?



## Updated: Processing Example 6.4

```
size(600, 300);
background(102);
fill(255);
noStroke();

for(int i = 60; i <= 120; i = i + 20)
{
   rect(50, i, 500, 10);
}
```



## For loop:  all parts are optional

```
for (  ;  ;  )
{
     // statements  here
}
```

This is an infinite loop…

## For loops can be nested

```
The value of i is: 0 and j is: 0
The value of i is: 0 and j is: 1
The value of i is: 0 and j is: 2
The value of i is: 0 and j is: 3
The value of i is: 1 and j is: 0
The value of i is: 1 and j is: 1
The value of i is: 1 and j is: 2
The value of i is: 1 and j is: 3
The value of i is: 2 and j is: 0
The value of i is: 2 and j is: 1
The value of i is: 2 and j is: 2
The value of i is: 2 and j is: 3
The value of i is: 3 and j is: 0
The value of i is: 3 and j is: 1
The value of i is: 3 and j is: 2
The value of i is: 3 and j is: 3
```

```
for (int i=0; i < 4; i++)
    for (int j=0; j < 4; j++)
        println("The value of i is: " + i + " and j is: " + j);
```

## Topics list

- Variable Scope
- Repetition in Programming (for loops).
- $\boxed{\text{Compound Assignment Statements}}$

## A note on i++

- The **post-body action** in this for loop is i++.
- This is called a compound assignment statement.
- It is a shortcut for i = i + 1.

```
for(int i = 0; i < 4; i++)
{
    rect(50, yCoordinate, 500, 10);
    yCoordinate = yCoordinate + 20;
}
```

# 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?