

Iteration in Programming

for loops

Produced Dr. Siobhán Drohan
by: Ms. Mairead Meagher



Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

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

Topics list

- There are three types of loop in programming:
 - while loops:
 - Counter controlled (n times) - covered in previous talk
 - Sentinel based (covered later in the course)
 - Flag based (covered later in the course)
 - for loops (this slide deck)
 - do While loops (covered later in the course)
- Comparative use of while and for loops
 - Lab03 - Challenge 1
 - Lab03 - Challenge 3

For loop pseudo-code

General form of a for loop

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

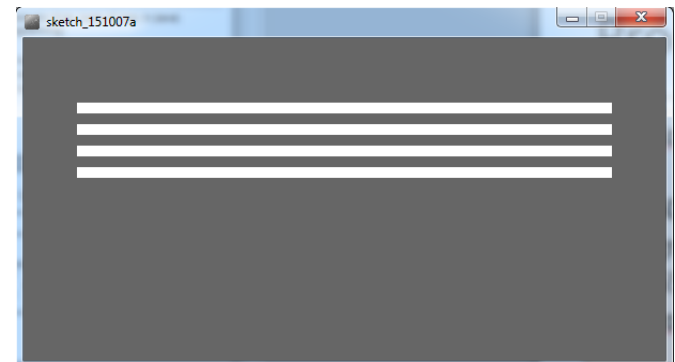
Recap: Processing Example 4.5

```
int yCoordinate = 60;

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

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

This was a slide from the previous talk. We used a while loop to repeatedly print the four rectangles to the display window.



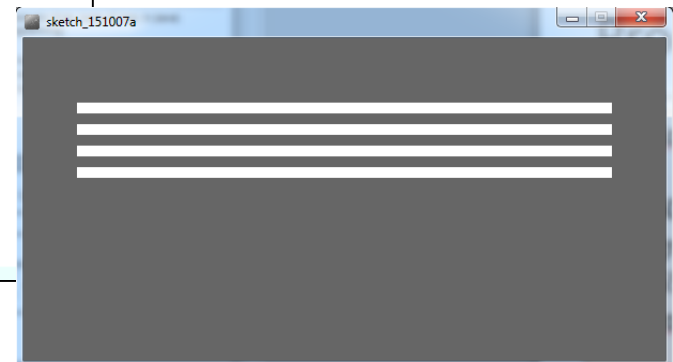
Processing Example 4.7

```
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;  
}
```

This code does the same as the previous slide, except that we use a different loop: **for**



For loop syntax

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

For loop syntax

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

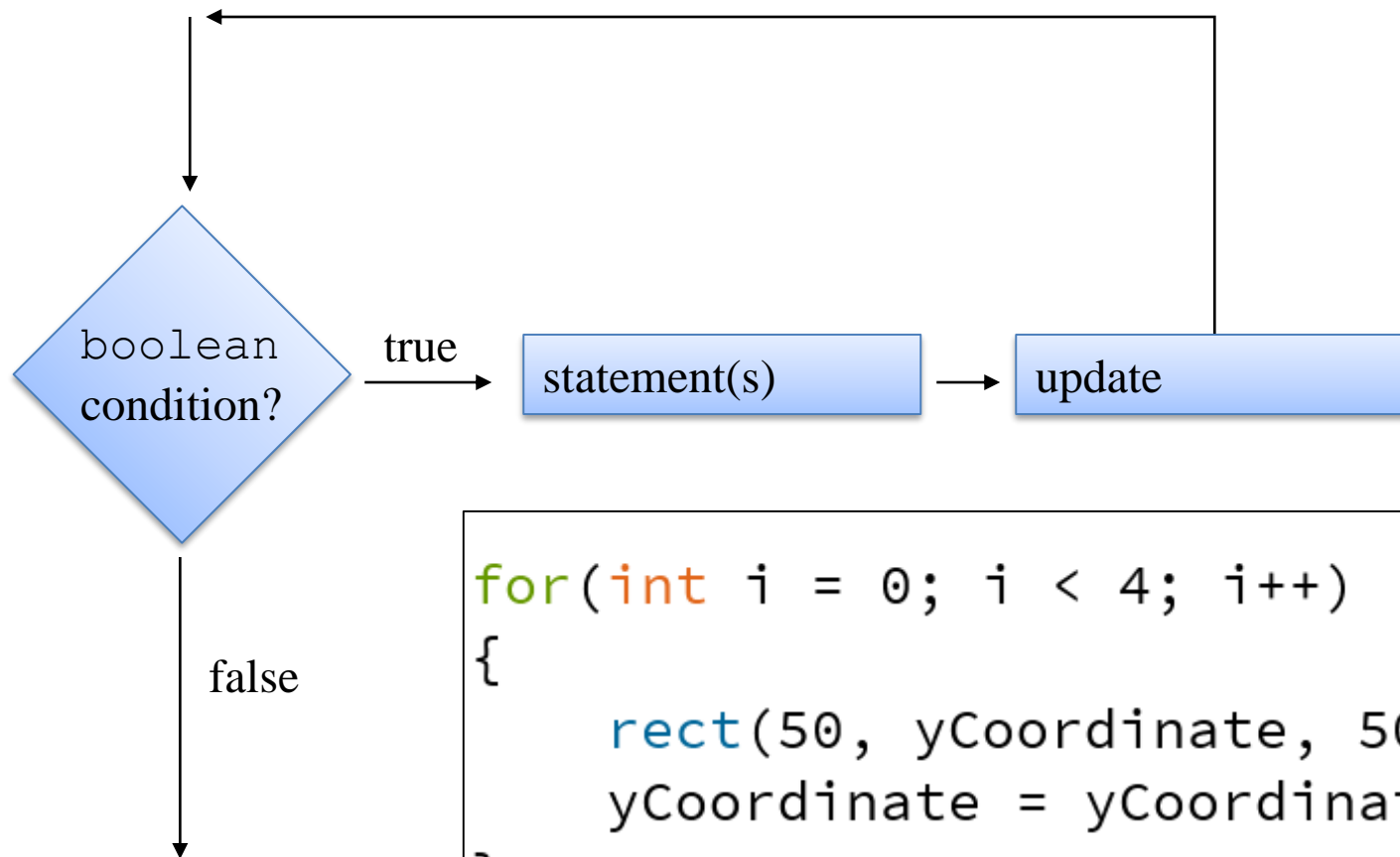
```
for(initialization; boolean condition; post-body action)  
{  
    statements to be repeated  
}
```

For loop syntax

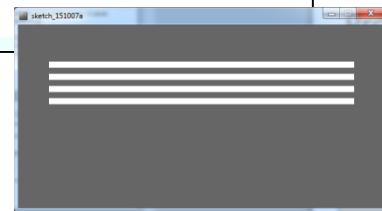
```
for(int i = 0; i < 4; i++)
```

initialization	<code>int i = 0;</code>	Initialise a loop control variable (LCV) e.g. i. It can include a variable declaration.
boolean condition	<code>i < 4;</code>	Is a valid boolean condition that typically tests the loop control variable (LCV).
post-body action	<code>i++</code>	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 4.7

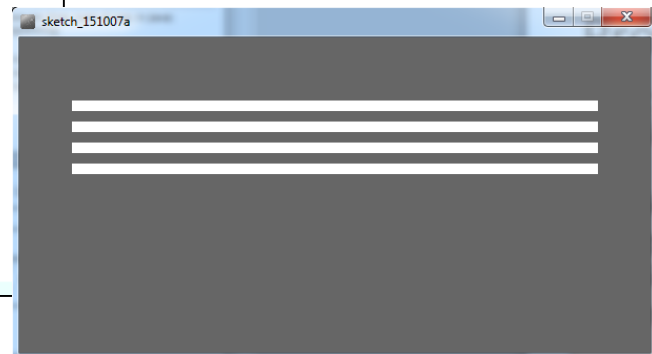
```
int yCoordinate = 60;

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

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

Q: Do we need the **yCoordinate** variable?

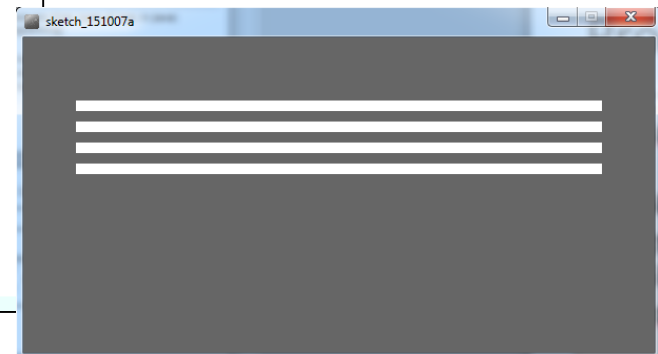
Can you think of a different approach using a **for** loop?



Processing Example 4.8

```
size(600, 300);  
background(102);  
fill(255);  
noStroke();  
  
for(int i = 60; i <= 120; i = i + 20)  
{  
    rect(50, i, 500, 10);  
}
```

A: We can eliminate the **yCoordinate** variable by setting the **i** variable to 60 and incrementing it by 20.



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

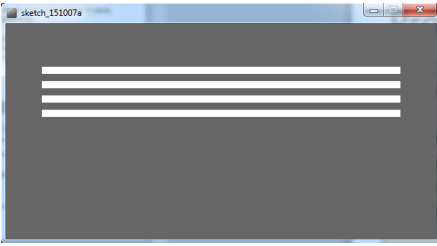
→ for (int i=0; i < 4; i++)

→ for (int j=0; j < 4; j++)

println("The value of i is: " + i + " and j is: " + j);

Topics list

- There are three types of loop in programming:
 - While loops:
 - Counter controlled (n times) - covered in previous talk
 - Sentinel based (covered later in the course)
 - Flag based (covered later in the course)
 - For loops (this slide deck)
 - Do While loops (covered later in the course)
- Comparative use of while and for loops
 - Lab03 - Challenge 1
 - Lab03 - Challenge 3



for versus while

Processing Example 4.7(for loop)

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

Processing Example 4.5 (while loop)

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

Variable **i** is the Loop Control Variable (**LCV**). It must be initialised, tested and changed.

int i = 0 is the **initialisation**.

i < 4 is the boolean condition i.e. the **test**

i++ is the post-body action i.e. the **change**.

Topics list

- There are three types of loop in programming:
 - While loops:
 - Counter controlled (n times) - covered in previous talk
 - Sentinel based (covered later in the course)
 - Flag based (covered later in the course)
 - For loops (this slide deck)
 - Do While loops (covered later in the course)
- Comparative use of while and for loops
 - Lab03 - Challenge 1
 - Lab03 - Challenge 3

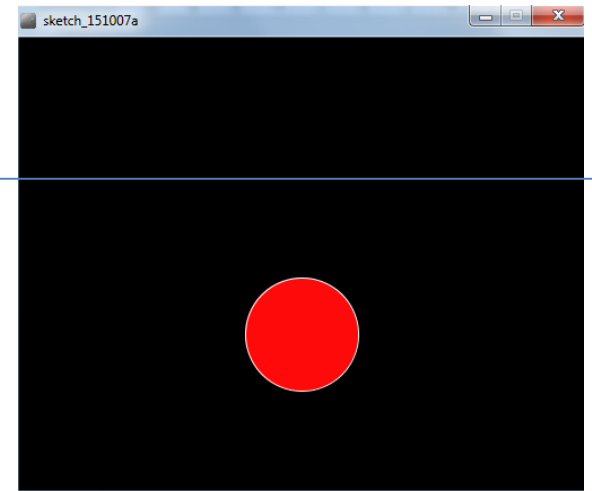
Lab03 - Challenge 1 – bouncing ball

Draw a continuously bouncing ball. (vertical only)

- the **xCoordinate** remains the **same** value
the **yCoordinate** will **change**.

Assumptions:

- display window is **500 x 400**
- ball is **100** in diameter.
- static **xCoordinate** is 250.
- **background** is called in the draw() method.
- starting **yCoordinate** is **300**.



Lab03 - Challenge 1

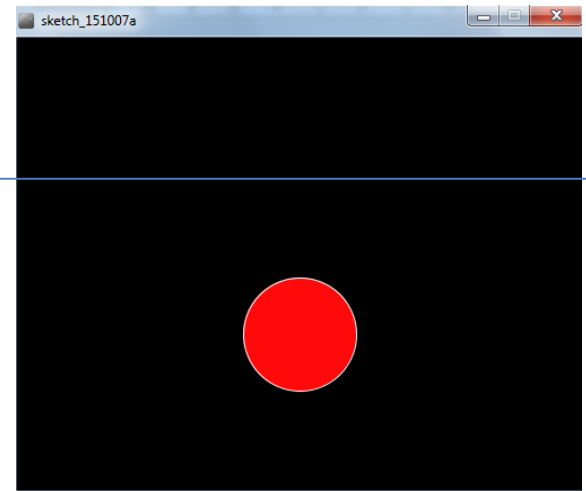
```
float yCoordinate = 300;
```

```
void setup() {  
  size(500,400);  
  fill(255, 10, 10);  
  stroke(255);  
}
```

```
void draw() {  
  background(0);  
  ellipse(250, yCoordinate, 100, 100);  
}
```

Assumptions:

- display window is **500 x 400**
- ball is **100** in diameter.
- static **xCoordinate** is 250.



Lab03 - Challenge 1

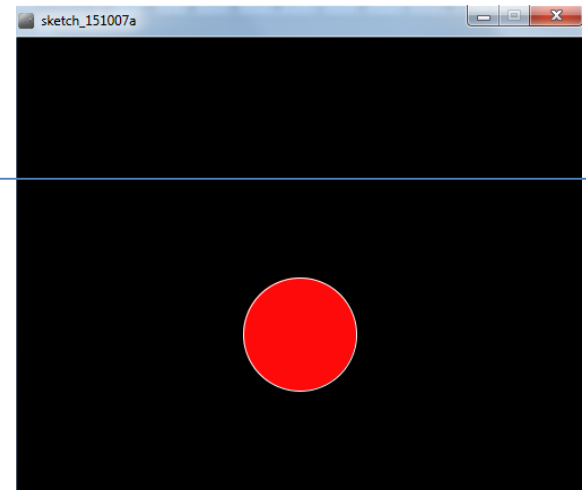
```
float yCoordinate = 300;
```

```
void setup() {  
  size(500,400);  
  fill(255, 10, 10);  
  stroke(255);  
}
```

```
void draw() {  
  background(0);  
  ellipse(250, yCoordinate, 100, 100);  
}
```

Assumptions:

- display window is **500 x 400**
- ball is **100** in diameter.
- static **xCoordinate** is 250.
- **background** is called in the draw() method.
- starting **yCoordinate** is **300**.



Lab03 - Challenge 1

```
float yCoordinate = 300;  
boolean bounceUp = false;  
  
void setup() {  
    size(500,400);  
    fill(255, 10, 10);  
    stroke(255);  
}
```

```
void draw() {  
    background(0);  
    ellipse(250, yCoordinate, 100, 100);  
  
    if (bounceUp)  
        // code to bounce the ball up  
  
    if (!bounceUp)  
        // code when ball is falling  
}
```

- We need to track whether the ball is bouncing up or falling.
- To do this, we will use a boolean variable **bounceUp**.
It will be:
 - **true** if the ball is bouncing up
 - **false** if the ball is falling and

```
float yCoordinate = 300;  
boolean bounceUp = false;
```

```
void setup() {  
  size(500,400);  
  fill(255, 10, 10);  
  stroke(255);  
}
```

```
void draw() {  
  background(0);  
  ellipse(250, yCoordinate, 100, 100);
```

```
  //ball is bouncing up  
  if (bounceUp){  
    if (yCoordinate > 100)  
      yCoordinate = yCoordinate - 1;  
    else  
      bounceUp = false;  
  }
```

```
  //ball is falling down  
  if (!bounceUp){  
    if (yCoordinate <= 350)  
      yCoordinate = yCoordinate + 1;  
    else  
      bounceUp = true;  
  }
```

```
}
```

Topics list

- There are three types of loop in programming:
 - While loops:
 - Counter controlled (n times) - covered in previous talk
 - Sentinel based (covered later in the course)
 - Flag based (covered later in the course)
 - For loops (this slide deck)
 - Do While loops (covered later in the course)
- Comparative use of while and for loops
 - Lab03 - Challenge 1
 - Lab03 - Challenge 3

Lab03 - Challenge 3 – Moving Line

- In a new sketch, draw a **vertical line** that is the height of your display window.
- It starts in the left most position of your display window and **moves right, pixel by pixel**, until it reaches the right hand side of your display window.

Lab03 - Challenge 3 – Moving Line

- Upon reaching the right hand side, the vertical line should **reverse direction** and return, pixel by pixel, to the left hand side of the display window.
- As your vertical line is continually traversing the display window, your **grayscale background should be varying** very slightly in colour.

Lab03 - Challenge 3 – Moving Line

Assumptions:

- Window size 300x400.
- Background is initially set to 120.
- Stroke weight is 4

```
float background = 120;
```

```
void setup()
```

```
{
```

```
    size(300,400);
```

```
    background(background);
```

```
    strokeWeight(4);
```

```
}
```



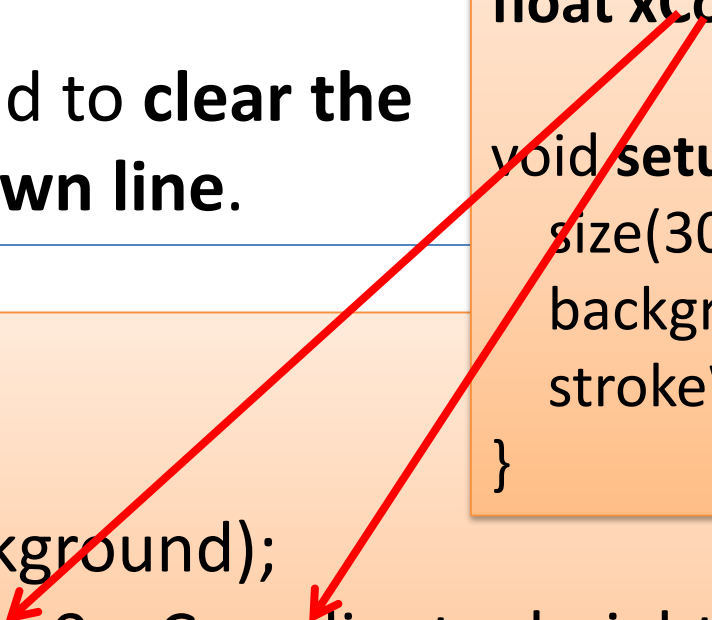
Lab03 - Challenge 3 – Moving Line

- Draw a **vertical line** that is the height of your display window.
- Call background to **clear the previously drawn line**.

```
void draw()
{
  background(background);
  line (xCoordinate, 0, xCoordinate, height);
}
```


```
float background = 120;
float xCoordinate = 0.0;

void setup(){
  size(300,400);
  background(background);
  strokeWeight(4);
}
```



Lab03 - Challenge 3 – Moving Line

This vertical line should start in the left most position of your display window and **move right, pixel by pixel**, until it reaches the right hand side of your display window.

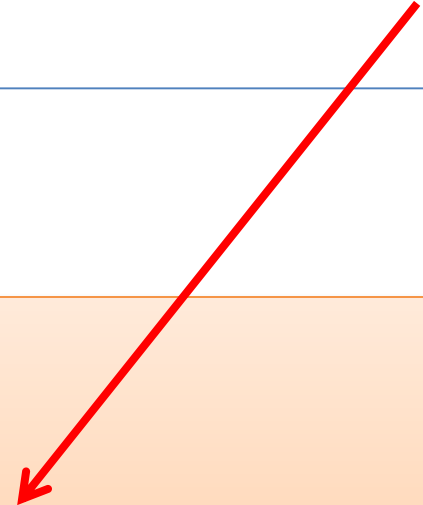


```
void draw(){  
    xCoordinate = xCoordinate + 1;  
    background(background);  
    line (xCoordinate, 0, xCoordinate, height);  
}
```

Lab03 - Challenge 3 – Moving Line

As your vertical line is continually traversing the display window, your **grayscale background** should be **varying** very slightly **in colour**.

```
void draw(){  
    xCoordinate = xCoordinate + 1;  
    background = background + 0.5;  
    background(background);  
    line (xCoordinate, 0, xCoordinate, height);  
}
```



Lab03 - Challenge 3 – Moving Line

- Upon reaching the right hand side, the vertical line should **reverse direction** and return, pixel by pixel, to the left hand side of the display window.
- We need to keep track of the direction that the line should be moving
i.e. is it going left-to-right, or has it reversed direction and gone from right-to-left?
- We will use a boolean variable to do this:
 - boolean **reverseDirection** will be initially set to false. indicating a left-to-right direction.
 - **false** indicates a **left-to-right direction**
 - **true** indicates a **right-to-left direction**.

Lab03 – Challenge 3

```
void draw()
{
  if (!reverseDirection){
    background = background + 0.5;
    xCoordinate = xCoordinate + 1;
  }
  else{
    background = background - 0.5;
    xCoordinate = xCoordinate - 1;
  }

  background(background);
  line (xCoordinate, 0, xCoordinate, height);
}
```

```
float background = 120;
float xCoordinate = 0.0;
boolean reverseDirection = false;
```

```
void setup(){
  size(300,400);
  background(background);
  strokeWeight(4);
}
```

Lab03 - Challenge 3 – Moving Line

- But, we have no code written that will set the flag to true e.g.

boolean reverseDirection = true;

- Under what circumstances should the flag be set to true?
- And when should it be set back to false?

```
void draw(){
```

```
  if (xCoordinate == width)
    reverseDirection = true;
  if (xCoordinate == 0)
    reverseDirection = false;
```

```
  if (!reverseDirection){
    background = background + 0.5;
    xCoordinate = xCoordinate + 1;
  }
  else{
    background = background - 0.5;
    xCoordinate = xCoordinate - 1;
  }
```

```
  background(background);
  line (xCoordinate, 0, xCoordinate, height);
}
```

```
float background = 120;
float xCoordinate = 0.0;
boolean reverseDirection = false;
```

```
void setup(){
  size(300,400);
  background(background);
  strokeWeight(4);
}
```


Questions?

