In the last lesson you learned …

- How to create a new Love program; create a folder and main.lua.
- How to write a bare bones Love program.
- That a function is a group of commands that perform a task.
- That a function can be given arguments to change how it behaves.
- That Love calls the love.update() function once when it starts.
- That Love calls love.draw() and love.update() in a loop until the program ends.

In this lesson you'll create a new program. Follow steps 1 to 3 in Chapter1 except that the folder will be called Proj02 instead of Proj01.

```
1  function love.load()
2  end
3
4  function love.draw()
5      love.graphics.print("Nina's Not a Ballerina",100,100)
6  end
7
8  function love.update()
9  end
10
```

This program will print the text in the Love window at the specified position of 100 pixels from the left and 100 pixels from the top. Let's learn how we can move this text around.

To move the text around we need to change the values of 100 for the horizontal (or X) position and 100 for the vertical (or Y) position in line 5. We'll do this using something called a variable.

In Lua, a variable is like an envelope that can contain anything. We give the variables names so we can tell them apart. Variable names in Lua can be any string of letters, digits, and underscores, not beginning with a digit.

```
VerticalPosition              OK
Vertical_Position             OK
Vertical Position             WRONG - contains space
VPos01                        OK
1VPos                         WRONG - starts with digit
```

Also you cannot use any of the Lua keywords as a variable name.

```
and       break     do        else      elseif
end       false     for       function  if
in        local     nil       not       or
repeat    return    then      true      until     while
```

Consider the following Lua instructions. The meaning of each is the text in gray.

```
XPos = 100              New value of XPos is 100
XInc = 2                New value of XInc is 2
XPos = XPos + 1         New value of XPos is the old value + 1 (or 101)
XPos = XPos + XInc      New value of XPos is the old value + 2 (or 103)
XInc = 5                New value of XInc is 5
XPos = XPos + XInc      New value of XPos is the old value + 5 (or 108)
```

We can use variables in place of numbers. By changing the value that we place in XInc, we can change the meaning of XPos = XPos + XInc from oldValue+2 to oldValue +5.

Let's put this to work in the sample program.

```
1  function love.load()
2      XPos = 100
3      YPos = 100
4  end
5
6  function love.draw()
7      love.graphics.print("Nina's Not a Ballerina",XPos,YPos)
8  end
9
10 function love.update()
11 end
```

This program produces the same output as the original but now the horizontal and vertical positions are stored in the variables XPos and YPos.

To move the printed text we can change the XPos and YPos values. The love.update() function is where we'll do this. It is called as soon as love.draw() is done.

```
1  function love.load()
2      XPos = 100
3      YPos = 100
4  end
5
6  function love.draw()
7      love.graphics.print("Nina's Not a Ballerina",XPos,YPos)
8  end
9
10 function love.update()
11     XPos = XPos + 1
12     YPos = YPos + 1
13 end
14
```

If you run this program, you'll see that each time the text is printed in love.draw() it is move 1 pixel to the right, and 1 pixel down because of what we do in love.update(). Eventually the text disappears out of the window never to return.

As long as our increments are written as 1 we cannot change this. So we will create two more variables called XInc and YInc.

```
function love.load()
    -- lines starting with -- are comments
    -- get the width and height of the Love window
    width = love.graphics.getWidth()
    height = love.graphics.getHeight()

    XPos = 100
    YPos = 100
    XInc = 1
    YInc = 1
end

function love.draw()
    love.graphics.print("Nina's Not a Ballerina",XPos,YPos)
end

function love.update()
    XPos = XPos + XInc
    YPos = YPos + YInc
end
```

In this new version of the program I added two instructions into love.load(). These two instructions ask Love how wide and tall its window is. If you run this program you see that the text behaves just as before.

Finally, lets tell Love what to do when the text is about to disappear off the screen.

Lua can perform a test and execute some instructions only if the test is true.

```
if something is true then
   execute these instructions
end
```

The **if** statement has more options than we're currently using. Here's an example.

```
if something is true then
   execute these instructions
elseif something else is true then
   execute these instructions
elseif something else is true then
   execute these instructions
else
   if no tests are true execute these instructions
end
```

All the **elseif** and **else** portions are optional and need be included only if you need them. More on this in a later tutorial.

```
function love.load()
    -- lines starting with -- are comments
    -- get the width and height of the Love window
    width = love.graphics.getWidth()
    height = love.graphics.getHeight()

    XPos = 100
    YPos = 100
    XInc = 1
    YInc = 1
end

function love.draw()
    love.graphics.print("Nina's Not a Ballerina",XPos,YPos)
end

function love.update()
    if XPos > width or XPos < 0 then
        XInc = -XInc
    end

    if YPos > height or YPos < 0 then
        YInc = -YInc
    end

    XPos = XPos + XInc
    YPos = YPos + YInc
end
```

In the final version of the program I have added a test. If the X position of the text is greater than the width of the window `(XPos > width)` then the text has disappeared off the right edge. Similarly, if the X position is less than zero `(XPos < 0)` then it's about to disappear off the left side. In either case I want to reverse the motion of the text. So I write

```
if XPos > width or XPos < 0 then
   execute these instructions
end
```

If either the first test is true **OR** the second test is true, then we should change the direction of the movement. If the increment is 1 it should become -1. And if it's -1 it should become 1. We write this as

```
XInc = -XInc

If XInc is 1 then the new value becomes -(1) or -1
If XInc is -1 then the new value becomes -(-1) or 1
```

The exact same case applies to the Y position except that we test against the height of the window and change YInc.

Now the text will bounce around inside the window. Try changing the values of XInc and YInc. You can make them bigger or smaller using decimal numbers like 0.5, 0.3, 0.015, etc.