Learning Objectives - For Loops

- **Explain `for` loop syntax**
- **Identify the causes of an infinite loop**
- **Identify the relationship between patterns, loops, and output**

For Loops

For Loop Syntax

Before you can start writing a loop, you need to be able to spot the pattern. Let’s take something simple:

```
System.out.println("Hello");
System.out.println("Hello");
System.out.println("Hello");
System.out.println("Hello");
System.out.println("Hello");
```

The pattern is `System.out.println("Hello");`, and it is repeated five times. Since we know that the loops needs to run exactly five times, a for loop is the way to go. Here is how you write a for loop that repeats five times. Use the code visualizer to see how a for loop works.

```
for(int i=0; i < 5; i++) {
    System.out.println("Hello");
}
```

Code Visualizer

Like **conditionals**, for loops are code blocks. In addition to a **boolean** statement, you declare, intialize and increment a **variable** called the loop *iterator*. All of the code that will be repeated needs to be between curly braces `{}`.

Understanding the loop header

Enter the code below and run it.

```
for(int i = 0; i < 5; i++) {
    System.out.println("Loop #" + i);
}
```

Code Visualizer

The loop ran five times, but the variable `i` did not start with 1. Instead it started with 0. Java, like most programming languages, starts counting with 0. Java will continue counting up to, but not including 5.

challenge

What happens if you:

- Change the print statement to `System.out.println("Loop #" + i + 1);`?
- Change the loop header to `i < 6` and the print statement to `System.out.println("Loop #" + i);`?
- Change the loop header to `i <= 5`?
- Change the loop header to `i--`?

Code Visualizer

▼ **Infinite Loops**

If you aren’t careful, you can wind up with an **infinite loop**. This means that you have a never ending loop. In the example above, if you change `i++` to `i–` then `i` will decrease, never reaching the value needed to make the boolean expression false.

Turtle Graphics

Before continuing with loops, we are going to learn how to create graphical output with the **Turtle Graphics** library. Like a pencil on paper, the Turtle object leaves a line as it moves around the screen.

Turtle Syntax

The first step is to create a Turtle object to move around the screen.

```
Turtle tina = new Turtle(); //creates a Turtle object called
        tina
```

Here are some basic commands to use with `tina` the Turtle object.

Command	Parameter	Description
`tina.forward(n)`	Where `n` represents the number of pixels	Move the turtle forward
`tina.backward(n)`	Where `n` represents the number of pixels	Move the turtle backward
`tina.right(d)`	Where `d` represents the number of degrees	Turn the turtle to the right
`tina.left(d)`	Where `d` represents the number of degrees	Turn the turtle to the left

Turtle Commands

Let's try this very simple command below. Copy it into the text editor on your left and then click the `TRY IT` button to see the graphical output.

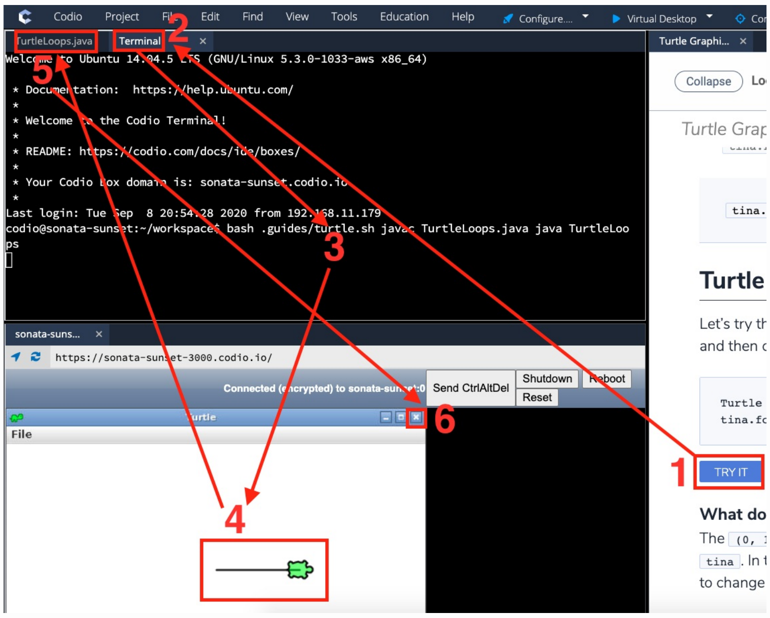
```
Turtle tina = new Turtle(0, 100); //change parameters to make
        tina visible
tina.forward(100);
```

What does the `(0, 100)` inside `Turtle()` do?

The `(0, 100)` inside `Turtle()` enables you to set the latitude and longitude of `tina`. In the example, `tina` starts at `0` pixel latitude and `100` pixels longitude. Feel free to change these parameters so that `tina` the Turtle can be seen on your screen.

Turtle Output

Below is an image highlighting what happens after the `TRY IT` button is clicked.



```
Welcome to Ubuntu 14.04.5 LTS (GNU/Linux 5.3.0-1033-aws x86_64)

 * Documentation:  https://help.ubuntu.com/
 *
 * Welcome to the Codio Terminal!
 *
 * README: https://codio.com/docs/ide/boxes/
 *
 * Your Codio Box domain is: sonata-sunset.codio.io
 *
Last login: Tue Sep  8 20:54:28 2020 from 192.168.11.179
codio@sonata-sunset:~/workspace$ bash .guides/turtle.sh javac TurtleLoops.java java TurtleLoo
ps
```

.guides/img/JavaTurtleOutput

1. `TRY IT` button is clicked by the user.
2. The `Terminal` tab is opened.
3. The terminal runs the command to compile the program and to display the graphical output.
4. The output is displayed as a canvas on the bottom left panel.
5. Click on the `TurtleLoops.java` tab to go back to the text editor if you want to make changes to the program.
6. Click on the `x` icon to close the canvas and exit the program. Alternatively, you can also press the `Ctrl` and `z` keys (Windows) or the `control` and `z` keys (Mac).

Turtle Coding - For Loop

Customize Your Turtle

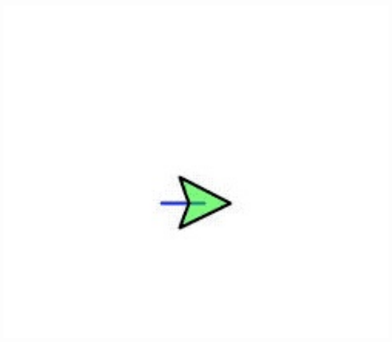
The following table provides additional commands you can use to customize `tina` the Turtle.

Command	Parameter	Examples
`tina.penColor("COLOR")`	Where `COLOR` represents the track or line color you want tina to leave behind	red, orange, yellow, green, blue, purple
`tina.shape("SHAPE")`	Where `SHAPE` represents the shape tina takes	turtle, circle, square, arrow, triangle
`tina.speed(s)`	Where `s` represents how many milliseconds it takes tina to perform an action	1 (fastest) through positive infinity (the larger the number, the slower tina moves)

Turtle Challenges

Now that you know how to customize `tina`, try to recreate the images you see below using your knowledge of `for` loops.

Challenge 1

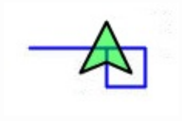


.guides/img/TurtleChallenge1

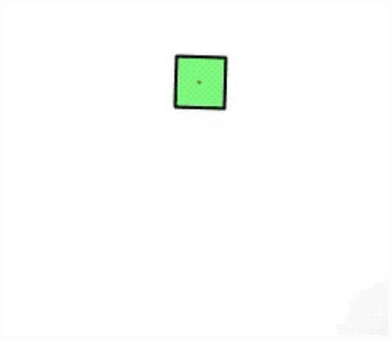
▼ **Hint**

There are multiple ways to accomplish this task but the trick lies within finding the **pattern** and then repeating it a **specific number of times**. One pattern in particular is to:

1. Go forward (creating a long line).
2. Make a right turn.
3. Go forward (creating a small line).
4. Make a right turn.
5. Go forward (creating another small line).
6. Make a right turn.
7. Go forward (creating a final small line).
8. Repeat steps #1 through #7 three more times for a total of **four** iterations.

The pattern should look something like this: 

Challenge 2

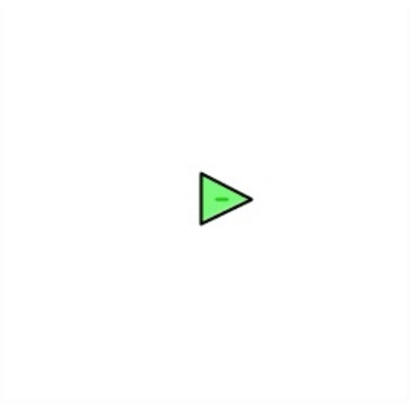


.guides/img/TurtleChallenge2

▼ **Hint**

Since a circle has 360 degrees, you will need a loop that repeats 360 times. Be careful about how far the turtle moves forward and turns. The circle can get very big, very quickly.

Challenge 3



.guides/img/TurtleChallenge3

▼ **Hint**

The pattern here is to move forward and make a right turn.

The trick lies within the fact that the distance the turtle moves has to get larger as the loop advances. Think of some operators that you can use to make the loop iterator variable get bigger during each iteration.

▼ **Still having trouble with creating the outputs above?**

Here are some sample solutions:

```
tina.penColor("blue");
tina.shape("arrow");
tina.speed(200);

for (int i = 0; i < 4; i++) {
  tina.forward(75);
  tina.right(90);
  tina.forward(25);
  tina.right(90);
  tina.forward(25);
  tina.right(90);
  tina.forward(25);
}
```

```
tina.penColor("red");
tina.shape("square");
tina.speed(10);

for (int i = 0; i < 360; i++) {
  tina.forward(1);
  tina.right(1);
}
```

```
tina.penColor("green");
tina.shape("triangle");
tina.speed(100);

for (int i = 10; i <= 200; i+=10) {
  tina.forward(i);
  tina.right(90);
}
```
