

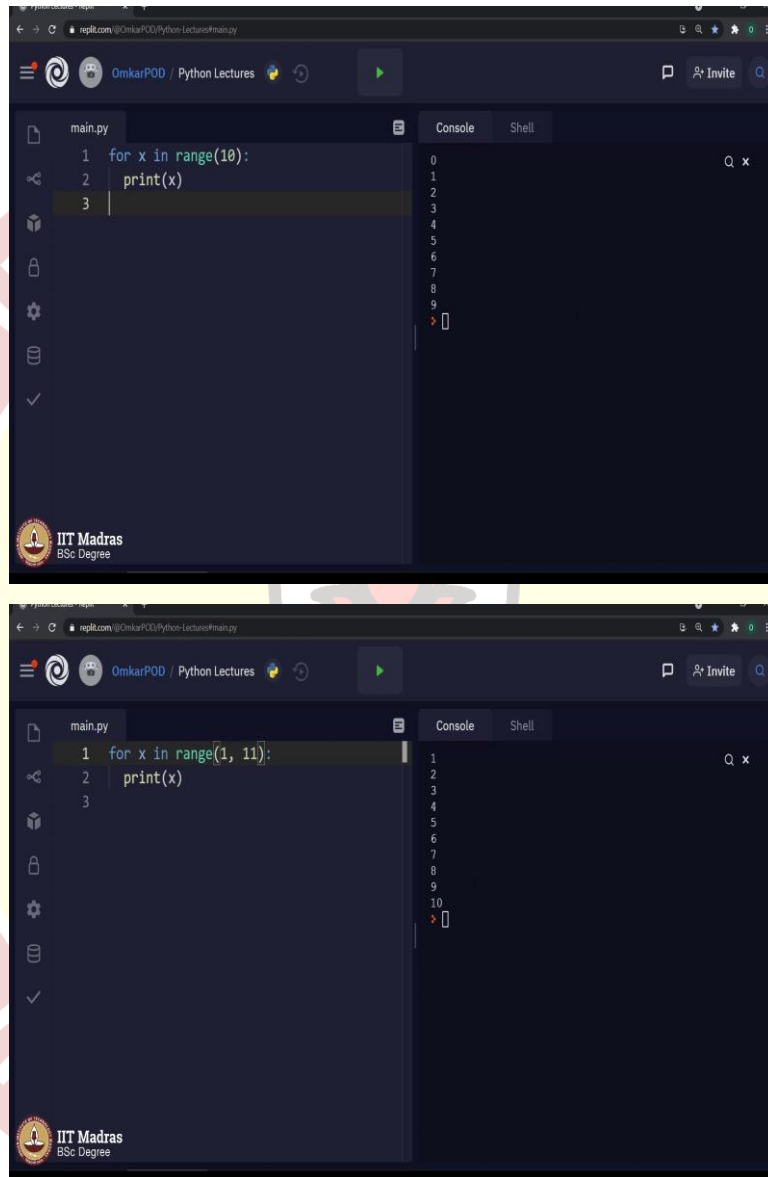


IIT Madras

ONLINE DEGREE

Programming in Python
Professor. Sudarshan Iyengar
Department of Computer Science and Engineering
Indian Institute of Technology, Ropar
More on range and for loop without range

(Refer Slide Time: 00:16)



The image contains two screenshots of a Python REPL interface, likely Repl.it, showing a for loop with range. The top screenshot shows the code `for x in range(10): print(x)` and the output `0 1 2 3 4 5 6 7 8 9`. The bottom screenshot shows the code `for x in range(1, 11): print(x)` and the output `1 2 3 4 5 6 7 8 9 10`. Both screenshots have a dark theme and a sidebar on the left with icons for file explorer, search, and settings. The bottom screenshot also has a search bar in the top right corner.

```
main.py
1 for x in range(10):
2   print(x)
3
```

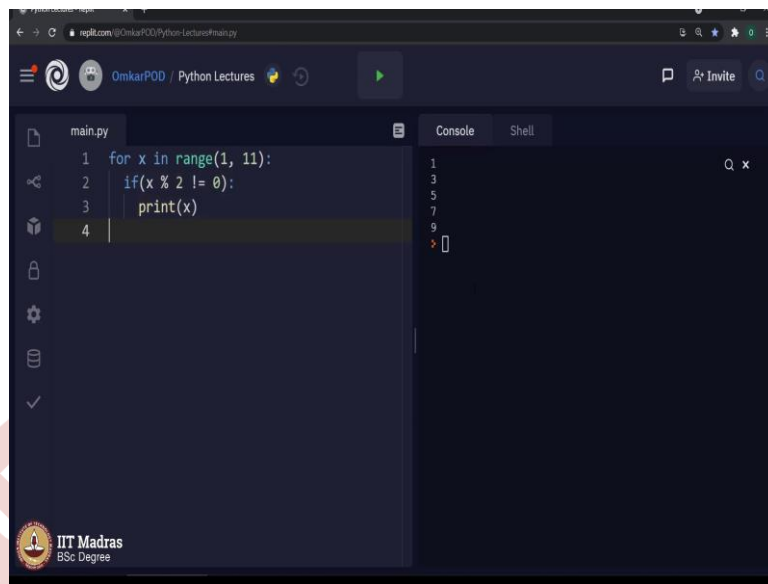
```
Console
0
1
2
3
4
5
6
7
8
9
>
```

```
main.py
1 for x in range(1, 11):
2   print(x)
3
```

```
Console
1
2
3
4
5
6
7
8
9
10
>
```

Hello, Python students. In this lecture we will discuss more about range. And also, we will see a different variation of for loop. Let us start with the range for x in range 10, print x. We have already seen a similar code, and you all know the output of this code, it will print numbers from 0 to 9. What if I want to print numbers from 1 to 10? Once again, you know the code, we will change it as 1, 11. It will print numbers from 1 to 10. So far, it is all good. Let me phrase one more question by adding one condition into this particular programme. I want all the odd numbers from 1 to 10.

(Refer Slide Time: 01:09)



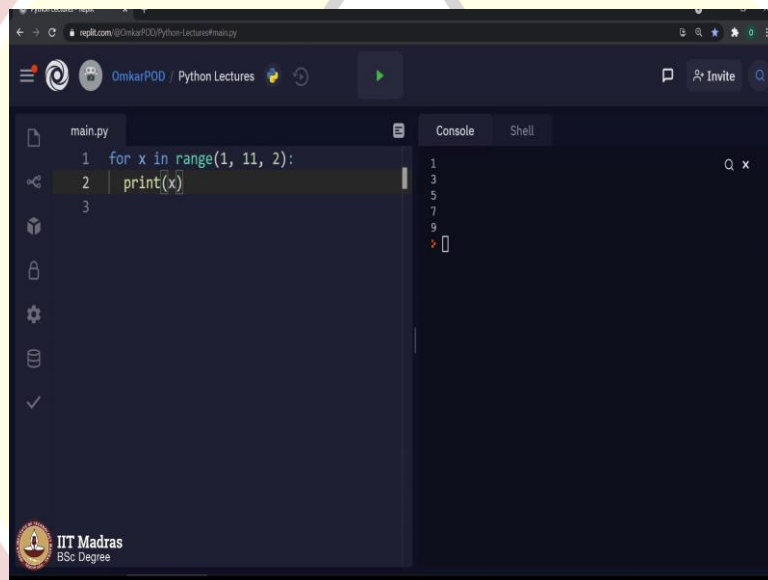
This screenshot shows a Python REPL interface with the following code in `main.py`:

```
1 for x in range(1, 11):  
2     if(x % 2 != 0):  
3         print(x)  
4
```

The console output displays the odd numbers from 1 to 10:

```
1  
3  
5  
7  
9  
>
```

The IIT Madras BSc Degree logo is visible in the bottom left corner.



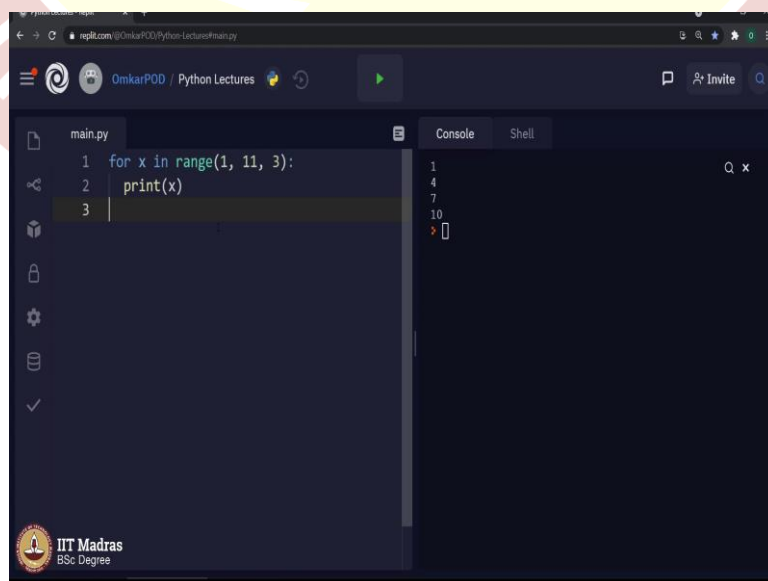
This screenshot shows a Python REPL interface with the following code in `main.py`:

```
1 for x in range(1, 11, 2):  
2     print(x)  
3
```

The console output displays the odd numbers from 1 to 10:

```
1  
3  
5  
7  
9  
>
```

The IIT Madras BSc Degree logo is visible in the bottom left corner.



This screenshot shows a Python REPL interface with the following code in `main.py`:

```
1 for x in range(1, 11, 3):  
2     print(x)  
3
```

The console output displays the numbers 1, 4, 7, and 10:

```
1  
4  
7  
10  
>
```

The IIT Madras BSc Degree logo is visible in the bottom left corner.

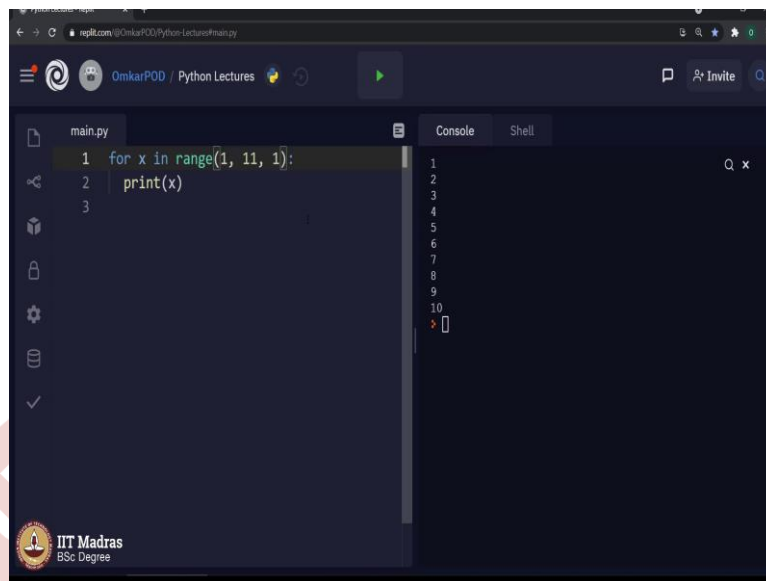
We can always write this if condition if $x \bmod 2 \neq 0$, print x . This will give us all the odd numbers between 1 to 10, which is absolutely correct. But instead of writing this if condition, there is alternate way with which we can change something over here in range and we can get the required odd numbers between 1 to 10. Let us see what that change is.

Let us remove this. And that change is, we can add one more parameter over here called step. In this case, we want only the alternate numbers between 1 to 10, which means that our step is plus 2. So, if we give value 2 over here, then it will print 1, then it will directly jump to 3, because the step is 2 then 5, then 7, and then 9, which is the expected output. Let us execute and see. As you can see, we are getting the expected output.

Now let us see what happens when we give these three parameters to this range. First parameter is considered as starting point, second parameter is considered as ending point and the third parameter is considered as step, which means computer will start from 1 and it will go till 11 minus 1, this is something we already know. But while incrementing from 1 to 11 minus 1, every time it will increment the value by 2, because that is the step value we have written over here.

Similarly, we can write it 3, then it will give 1 4 7 and 10 as expected. This time, it once again started from 1, but now instead of making it 2 the computer, will start from 1 and then for a second iteration, it will increment the value by 3. So, 1 plus 3, it will, you will get 4. Next time again plus 3, again plus 3. This is how the complete range function executes, as in range has three parameters all the time. Even if we provide only two values or even one value, internally, it considers three values.

(Refer Slide Time: 04:07)



The screenshot shows a web-based Python REPL interface. The code editor on the left contains the following Python code:

```
1 for x in range(1, 11, 1):  
2     print(x)  
3
```

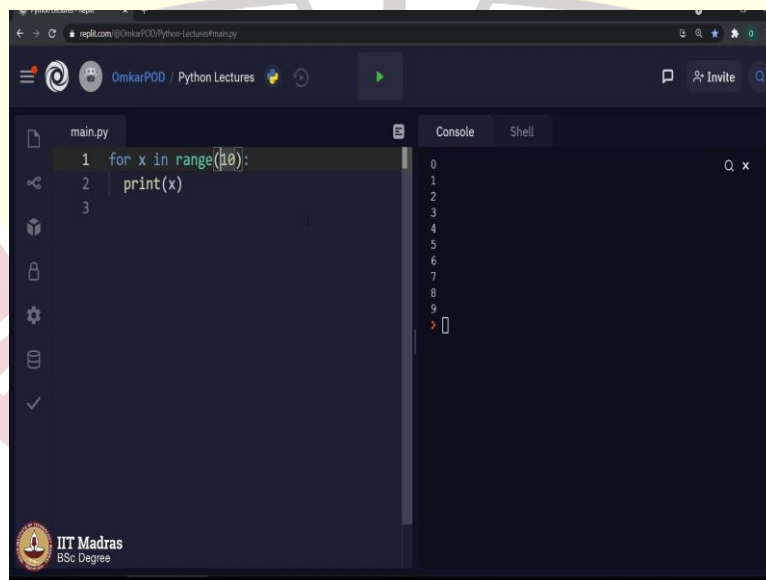
The 'Console' tab on the right displays the output of the code, which is the numbers 1 through 10, each on a new line:

```
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
>
```

The interface includes a file explorer on the left, a search bar, and a footer with the IIT Madras BSc Degree logo.

Now you must be wondering how that happens. For example, earlier, we were executing code, which was printing numbers from 1 to 10. In this case, the computer was taking the default value for that third parameter's step. And the default value for that parameter is 1. Now still it will print the same output.

(Refer Slide Time: 04:24)



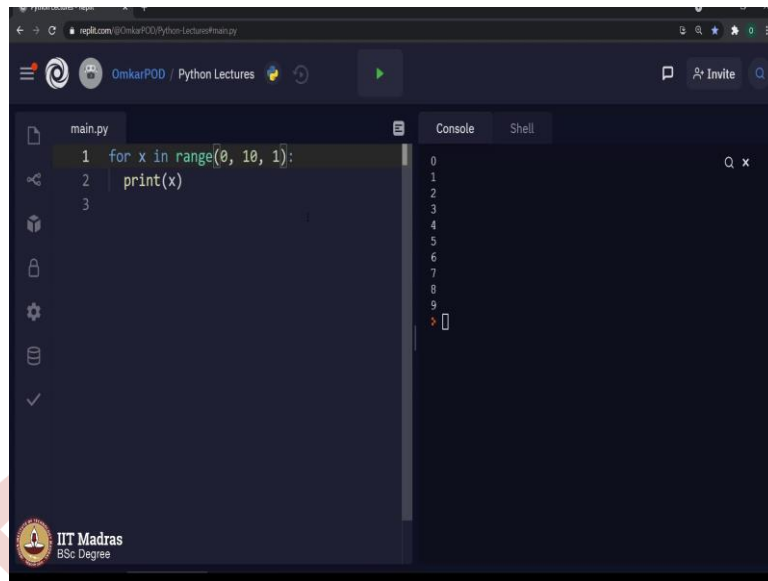
The screenshot shows the same web-based Python REPL interface. The code editor on the left contains the following Python code:

```
1 for x in range(10):  
2     print(x)  
3
```

The 'Console' tab on the right displays the output of the code, which is the numbers 0 through 9, each on a new line:

```
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
>
```

The interface includes a file explorer on the left, a search bar, and a footer with the IIT Madras BSc Degree logo.



The screenshot shows a web-based Python IDE interface. The top bar includes the Replit logo, the username 'OmkarPOD', and the project name 'Python Lectures'. Below the bar, the code editor displays a file named 'main.py' with the following Python code:

```
1 for x in range(0, 10, 1):  
2     print(x)  
3
```

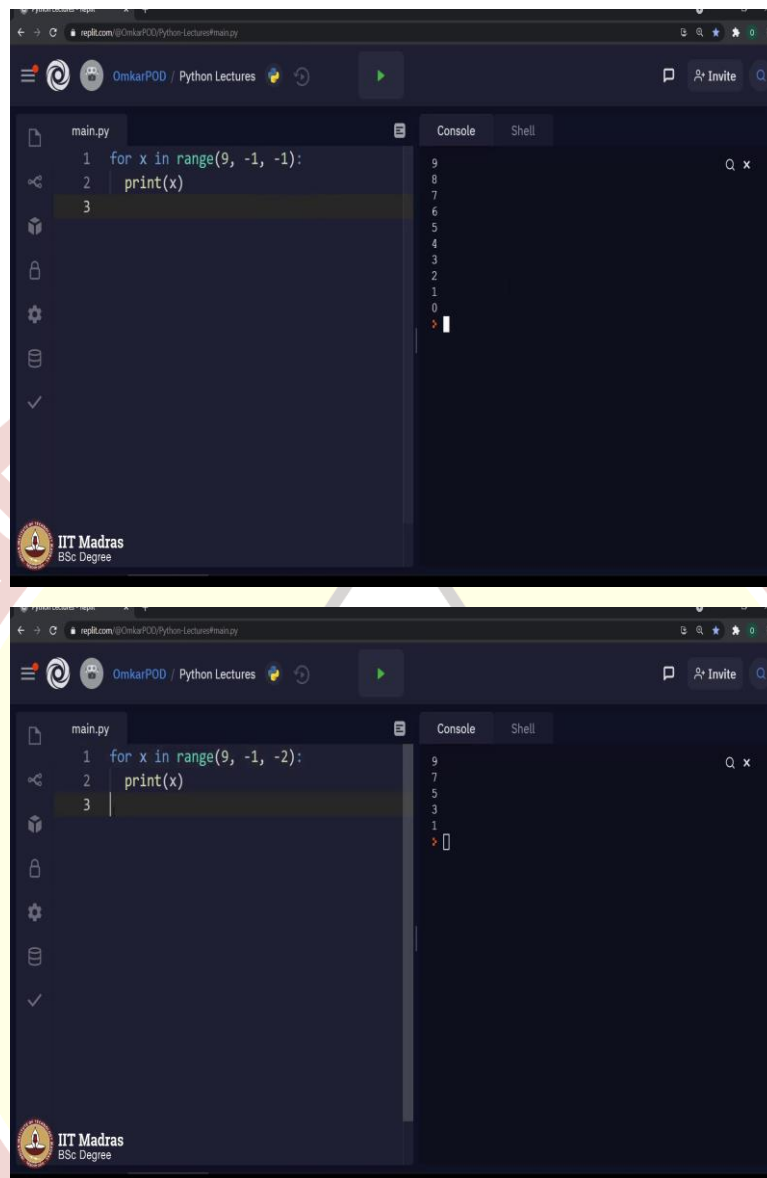
To the right of the code editor is a 'Console' tab, which shows the output of the program: the integers from 0 to 9, each on a new line. The bottom of the interface features a watermark for 'IIT Madras BSc Degree'.

Similarly, when we were executing a code like this with a single parameter, at that time, the output was numbers from 0 to 9. In this case, computer was considering the default value for start, which is first parameter as 0. And once again, the default value for step as 1, you will get the same output.

Based on these experiments, we can comfortably say that range has three parameters. First, is start, it is an optional parameter and its default value is 0. Second is end, it is a mandatory parameter, hence it do not have any default value. And the third one is step. Again, it is optional parameter. And its default value is 1.

But what if I want to print numbers in a reverse order, in a decreasing order? Let us say right now, I am printing numbers from 0 to 9. Instead of that, what if I want to print numbers from 9 to 0? Is it possible using for loop, using this range? Yes, of course, it is possible. We have to make small change in the parameters of range.

(Refer Slide Time: 05:54)



The image contains two screenshots of a Replit Python environment. The top screenshot shows a Python script with the following code:

```
1 for x in range(9, -1, -1):  
2     print(x)  
3
```

The console output shows the numbers 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, printed in descending order. The bottom screenshot shows the same script but with the range function modified to `range(9, -1, -2)`:

```
1 for x in range(9, -1, -2):  
2     print(x)  
3
```

The console output shows the numbers 9, 7, 5, 3, 1, printed in descending order, skipping even numbers.

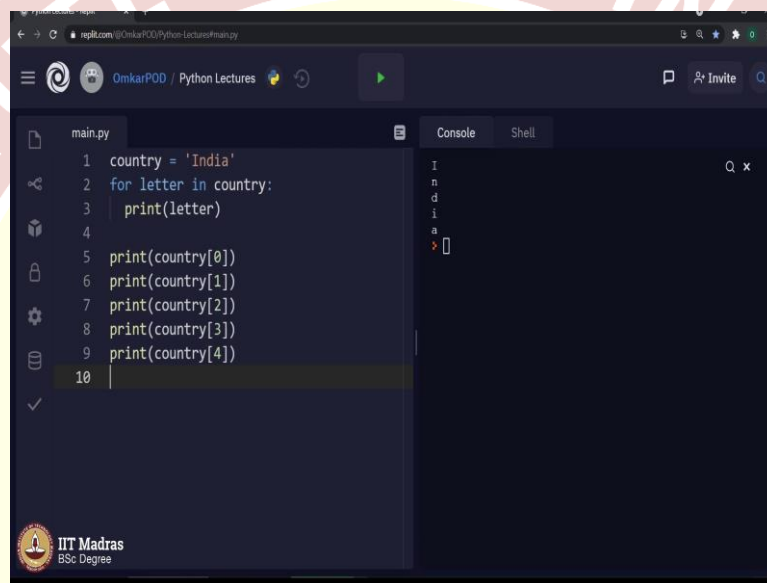
If we want to print numbers from 9 to 0, then starting point has to be 9. What about the ending point? We know always the end should be one number higher than the place where we want to stop our iteration. Now, the expected iteration is supposed to bring 9 8 7 6 5 4 3 2 1 0, that means we are supposed to stop at 0. In this particular range, what comes after 0? That is minus 1. Hence, it will be minus 1.

What about step? In this case, we are not incrementing, we are actually decrementing the value. Hence, step should be minus 1. What do you think? This particular code will give us the expected output or not? Let us try. As you can see, we are getting numbers from 9 to 0 as expected. Similarly, we can again make it minus 2 to get all odd numbers in the reverse order.

Therefore, we can conclude that range has three parameters and same range can be used to get values in descending order as well.

Moving to next point, which is for loop without range. Now, you must be wondering how that is even possible? So far, we have seen so many different examples of for loop, but all those examples use this particular range. Then how it is possible to write a for loop without using range. And the answer is, there is a way to write a for loop without range, which is sometimes called as for each, you have seen a similar for loop in computational thinking.

(Refer Slide Time: 07:56)

A screenshot of a web-based Python REPL interface. The left pane shows a file named 'main.py' with the following code:

```
1 country = 'India'
2 for letter in country:
3     print(letter)
4
5 print(country[0])
6 print(country[1])
7 print(country[2])
8 print(country[3])
9 print(country[4])
10
```

The right pane, labeled 'Console', shows the output of the code execution: 'I', 'n', 'd', 'i', 'a', each on a new line. The interface includes a search bar, a 'Run' button, and a 'Shell' tab. A watermark for 'IIT Madras BSc Degree' is visible in the bottom left corner.

Let us try that, let us look at this particular code. We have declared one variable called country and the value is India. Then for letter in country, print letter. First let us execute the code and then we will see how exactly it works. It is printing the entire string one character at a time. So, how exactly it works?

This particular for loop goes through the entire string, which is stored in variable country, character by character, which means the first value which is stored in variable letter is capital I. For next iteration, the value for variable letter will be updated to n, in third iteration it will become d, in fourth iteration, it will become i, in fifth iteration it will become a, as there is no letter after a in this particular variable, that is the place the loop will stop.

This code is similar like printing these print statements, where we are saying country of 0, country of 1, country of 2, 3 and 4. Country of 0 will print I, country of 1 will print n and so on till a. These 5 lines of code can be converted into 2 simple lines using this special feature

of for loop which is called as for each. For loop will execute for each letter or a character in the given string.

It is very easier to write a code like this and also it has many different advantages which we will explore in upcoming weeks. Thank you for watching this lecture. Happy learning.

