IIT Madras

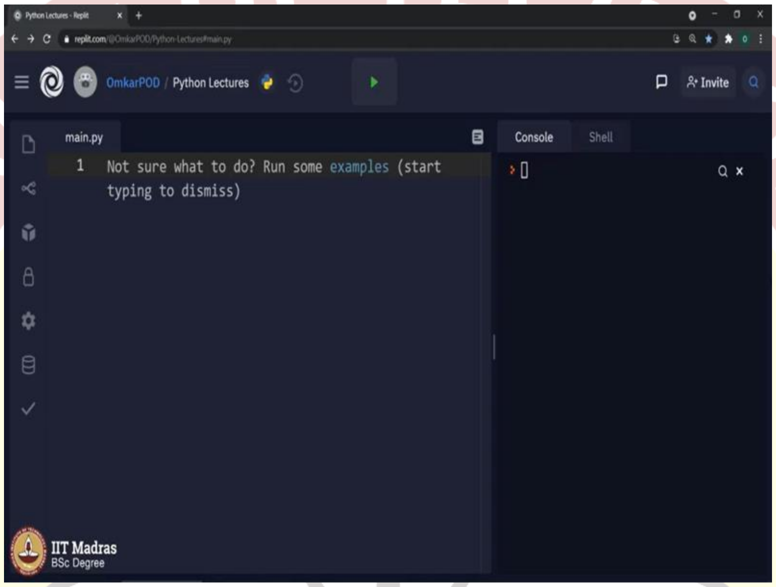
ONLINE DEGREE

Programming in Python

Professor. Sudarshan Iyengar
Department of Computer Science and Engineering
Indian Institute of Technology, Ropar
Mister. Omkar Joshi
Indian Institute of Technology, Madras

Formatted Printing

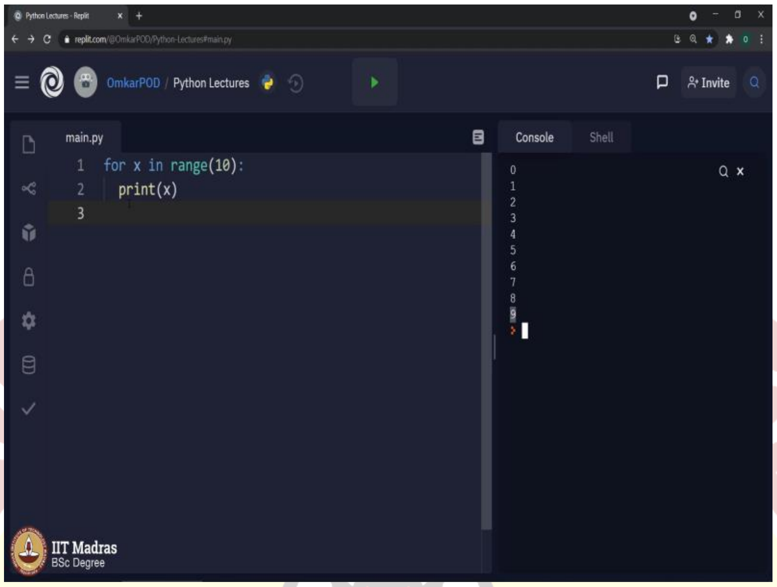
(Refer Slide Time: 00:16)


Python Lectures - Replit
replit.com/@OmkarPOD/Python-Lectures#main.py
OmkarPOD / Python Lectures
Invite
main.py
Console
Shell
IIT Madras
BSc Degree

```
Not sure what to do? Run some examples (start
typing to dismiss)
```

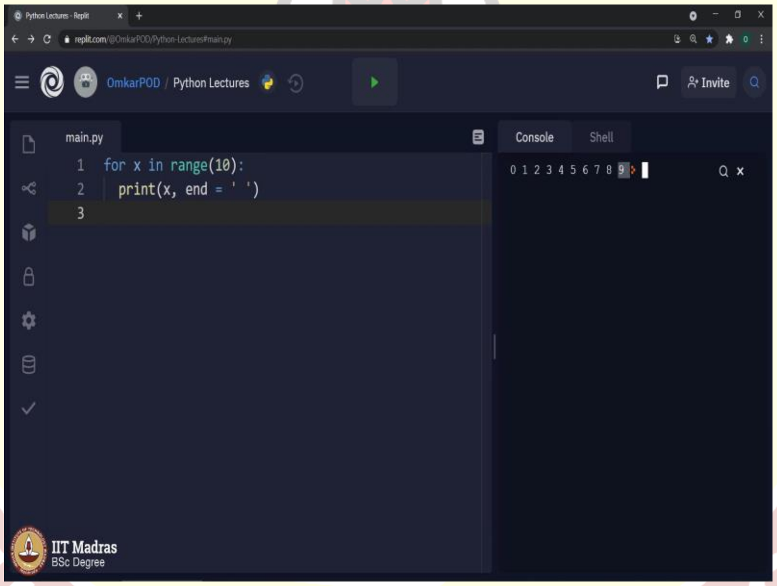
Hello Python students. At this point of time, we all are very much comfortable with print statement. Therefore, in this lecture, we will introduce few more features of the same print statement. We will discuss these print features along with loops because these features will allow us lot of flexibility when we use print statement inside loop.

(Refer Slide Time: 00:48)



```
for x in range(10):
    print(x)
```

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



```
for x in range(10):
    print(x, end = ' ')
```

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

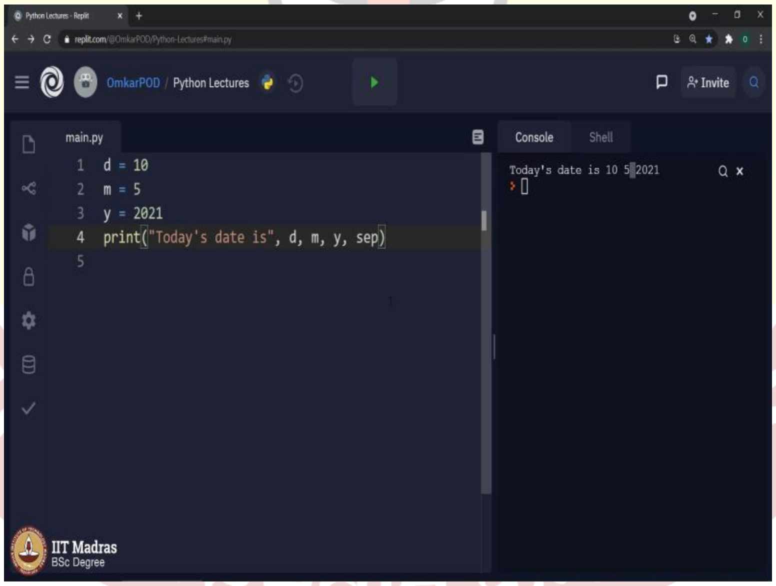
Let us look at the first feature. Let us see this particular program. We all know what exactly the output is going to be. It will print all the numbers from 0 to 9. But if you have noticed, while printing these numbers, this particular print statement is printing each number on a new line because that is how this print works. Every time computer comes across a print statement, it goes to a new line and executes the print statement. Therefore, the first value will be printed on the first line.

And whenever computer comes to this print once again, which is when the value of x is 1, it will go to the next line and print it on the second line and so on till we reach to 9. But, what if I want to print these numbers in a single line? Which means, I want to tell computer that whenever you come across print, do not go to next line, I want you to stay on the same line and print all these numbers in a single line. It is possible using a feature called end, end let us say, space. Let us execute this code. As you can see, the same output, we are getting but now all the numbers are getting printed in a single line.

Now, let us see how exactly it works. The default value for this end parameter of print is backslash n, which is new line. But in this case, we are explicitly telling the computer that override that default value and consider this space as a new value for end. Which means, computer will print x and end with a space. Therefore, in first iteration, computer will print 0 and add space.

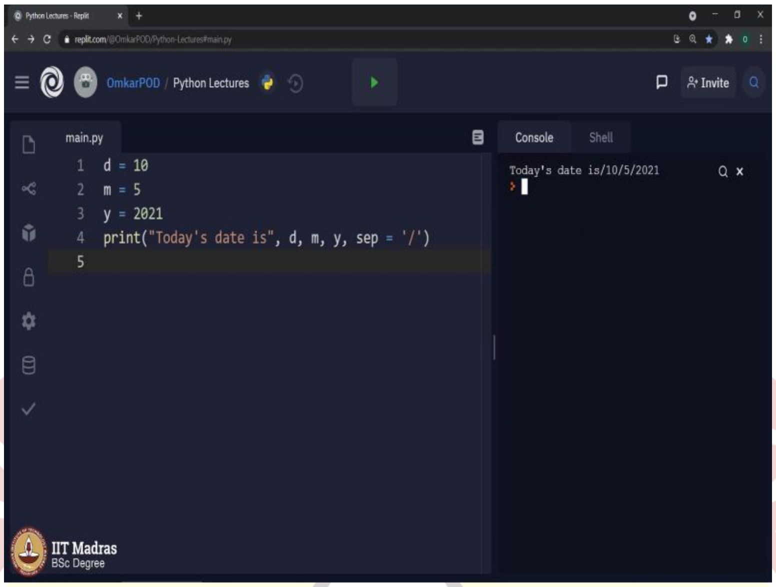
In second iteration, computer will print 1 and add space, which means computer will print x and end with a space. The same thing will be continued till 9 followed by space. Now, let us look at one more similar parameter of this print statement.

(Refer Slide Time: 03:23)


Python Lectures - Replit
replit.com/@OmkarPOD/Python-Lectures#main.py
OmkarPOD / Python Lectures
Invite
main.py
Console
Shell
IIT Madras
BSc Degree

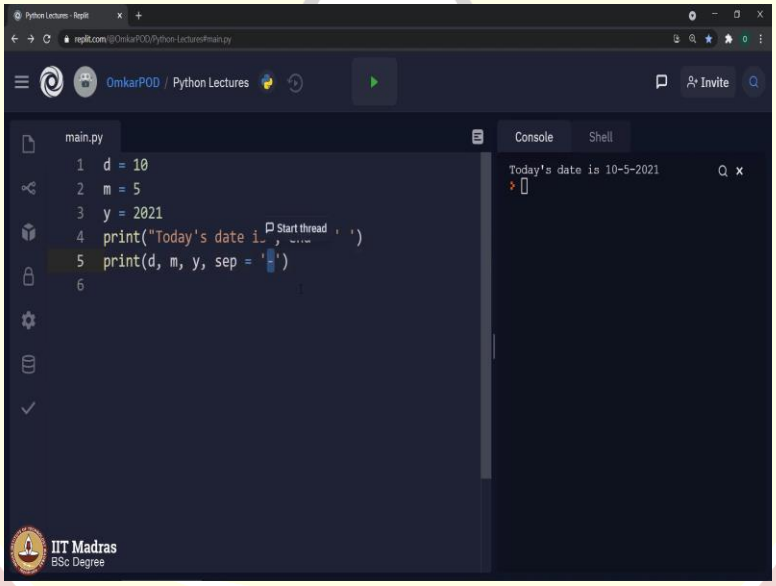
```python
d = 10
m = 5
y = 2021
print("Today's date is", d, m, y, sep)
```

```
Today's date is 10 5 2021
> 
```



```python
d = 10
m = 5
y = 2021
print("Today's date is", d, m, y, sep = '/')
```

```
Today's date is/10/5/2021
>
```



```python
d = 10
m = 5
y = 2021
print("Today's date is", end = ' ')
print(d, m, y, sep = '-')
```

```
Today's date is 10-5-2021
>
```

Let us look at this particular code. We all know what is going to be the output of this particular program. Today's date is 10, 5, 2021. But as you can see over here, this is not the way we write a date. We always put either a hyphen or a slash in between this day, month and year. Now, the question is, how to write this particular date in any of those formats?

In order to answer this question, first we have to figure out why this particular space is getting added between these numbers, even though we are not doing it over here. But still, computer is adding space between is and 10, 10 and 5 and then 5 and 2021. This thing is happening because the print statement has one more parameter which is referred as sep, which stands for separator. And the default value for this separator is space.

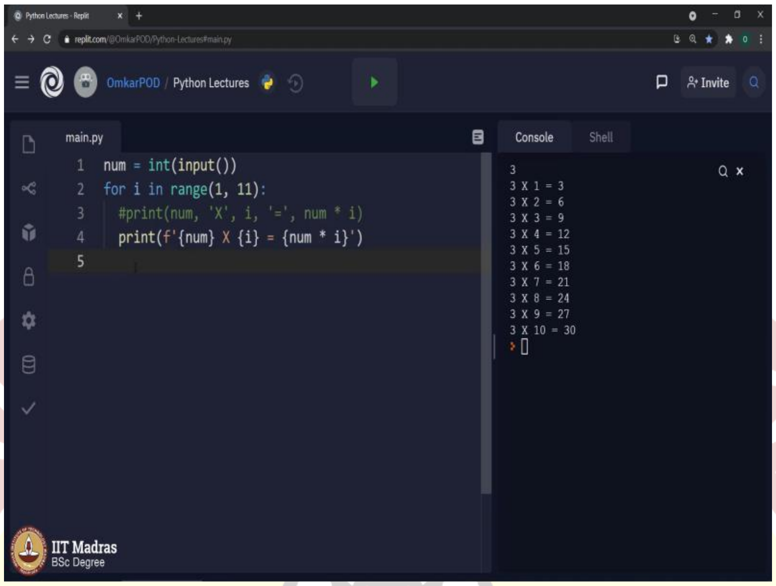
Because of that, whenever we use comma in order to separate multiple values in a single print statement, computer separates these values using these spaces. Once again, we can overwrite the default value of this sep parameter by explicitly mentioning some value, for example, a slash. Let us execute and see what happens.

Now, we got the expected value; today's date 10 slash 5 slash 2021. But still, there is a small error. Now, in between is and 10, there is a slash. We do not want slash over there, we want space. But this time, we are explicitly saying a separator has to be slash. Then, how we can remove this slash between is and 10? It is simple. We can write a new print statement and over here, we print d, m, y and we will specify separator as slash. We will remove it from here.

Now, let us see what happens in this case. We got the output, but once again, the date is on the next line. This is happening because, correct, we saw something called as end just few minutes back. So, we will make end is equal to space. Today's date is 10 slash 5 slash 2021. Now, that is the perfect way to write a date. If we want a hyphen, we can replace this slash by hyphen. 10 dash 5 dash 2021. We can use any of these formats by mentioning what should be the separating character.

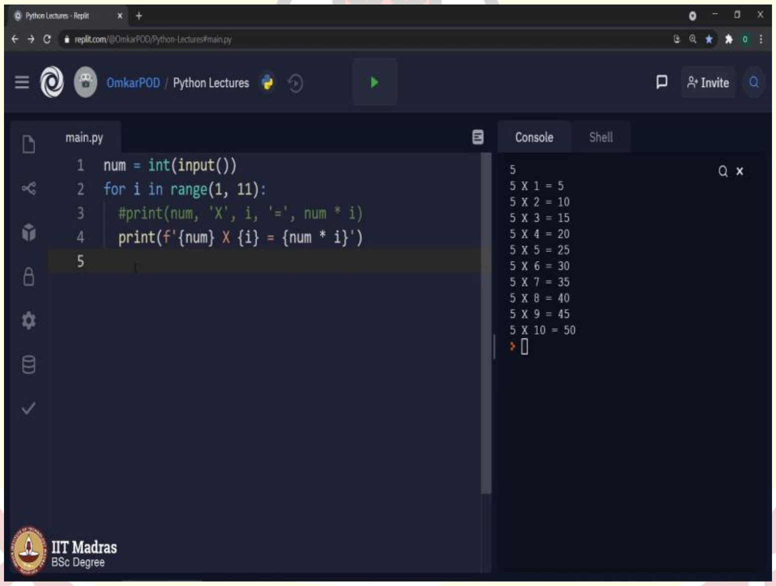
These two parameters end and separate of print statements, comes very handy when we use this print inside a loop where we may not want to print the values every time on a new line. Or we may want to separate those values using a specific character. Let us move to next feature of this print statement.

(Refer Slide Time: 07:10)



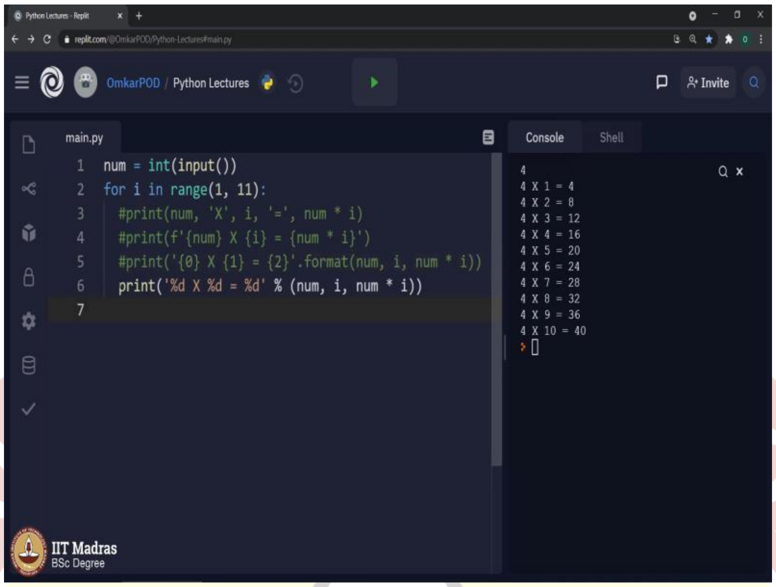
```
num = int(input())
for i in range(1, 11):
    #print(num, 'X', i, '=', num * i)
    print(f'{num} X {i} = {num * i}')
```

```
3
3 X 1 = 3
3 X 2 = 6
3 X 3 = 9
3 X 4 = 12
3 X 5 = 15
3 X 6 = 18
3 X 7 = 21
3 X 8 = 24
3 X 9 = 27
3 X 10 = 30
>
```



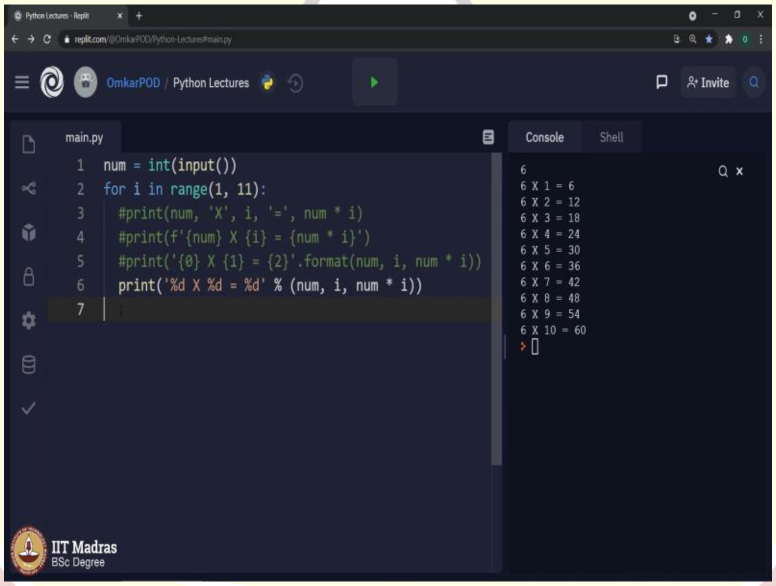
```
num = int(input())
for i in range(1, 11):
    #print(num, 'X', i, '=', num * i)
    print(f'{num} X {i} = {num * i}')
```

```
5
5 X 1 = 5
5 X 2 = 10
5 X 3 = 15
5 X 4 = 20
5 X 5 = 25
5 X 6 = 30
5 X 7 = 35
5 X 8 = 40
5 X 9 = 45
5 X 10 = 50
>
```



```
num = int(input())
for i in range(1, 11):
    #print(num, 'X', i, '=', num * i)
    #print(f'{num} X {i} = {num * i}')
    #print('{0} X {1} = {2}'.format(num, i, num * i))
    print('%d X %d = %d' % (num, i, num * i))
```

```
4
4 X 1 = 4
4 X 2 = 8
4 X 3 = 12
4 X 4 = 16
4 X 5 = 20
4 X 6 = 24
4 X 7 = 28
4 X 8 = 32
4 X 9 = 36
4 X 10 = 40
```



```
num = int(input())
for i in range(1, 11):
    #print(num, 'X', i, '=', num * i)
    #print(f'{num} X {i} = {num * i}')
    #print('{0} X {1} = {2}'.format(num, i, num * i))
    print('%d X %d = %d' % (num, i, num * i))
```

```
6
6 X 1 = 6
6 X 2 = 12
6 X 3 = 18
6 X 4 = 24
6 X 5 = 30
6 X 6 = 36
6 X 7 = 42
6 X 8 = 48
6 X 9 = 54
6 X 10 = 60
```

Let us look at this particular code. Here, we are accepting a number from the user and then, we are running the loop from 1 to 10, num multiplied by i equal to num multiplied by 10. Can you guess what will be the output of this program? Correct, it will print a multiplication table of the input number. Let us say 3, it will print the numbers from 3 to 30. There is nothing new about this, we have already seen it before.

So, this time, we will introduce a different variant of this particular print statement in order to print the same output in an easier manner. Let us say, how we can replace this print with something else?

Let us look at this particular line 4. Let me comment this line. Now, look at this particular line, number 4. Here, we are using something called as f. This stands for f print, which means formatted printing. Here, the entire statement will be written as a single string instead of printing multiple values in a single print. In earlier case, we were using number, which is a integer, then a string, then another integer, then a string and then another integer.

So, over there, we were printing 1, 2, 3, 4 and 5 values in a single statement. But instead of that, here, we are printing a single string value and inside that string value, we are using those variables. Because of this character f, over here, computer will realize it is a formatted print statement. Hence, whatever the value which is mentioned inside these parenthesis, as in, inside these curly brackets will be considered as variable, rest everything will be considered as string.

Therefore, while executing this particular program, computer will replace this variable num with its actual value. Same for i and same for num multiplied by i. Let us execute and see the output. Let us say 5, we got the multiplication table in that same format. There is one more way to achieve a similar output using a different variation of this print statement. Let us see how that works.

Look at this line number 5, once again, we are printing only one string here, followed by something new dot format. This kind of print statement is called as print using format function which means we are simply printing these values over here. 0, 1 and 2 refers to these three variables over here. 0 is this num. 1 is this i and 2 is the computed value of num multiplied by i.

Because of this format function, computer will execute this code and each this value which is 0, 1 and 2 will be replaced by its equivalent value stored in these variables num, i and num multiplied by i. Let us execute and observe the output. 4, we got the multiplication table in the same format. In addition to this, there is one more way through which we can write the same print statement.

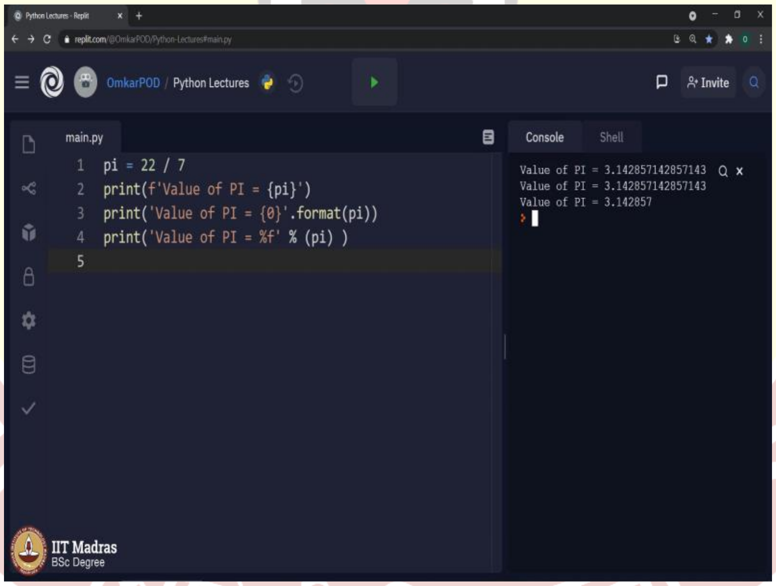
Let us comment this. Now, look at this line number 6. This is an old way of writing print statement. C programing language used to follow this particular style of writing the print statement, therefore, Python also supports this particular way of writing print statement. Let us see how it works.

Percent d, d stands for integer because number is an integer variable. Percent d will be replaced by num. Second percent d will be replaced by i and third percent d will be replaced by num multiplied by i. This particular way of writing a print statement is referred as print using string modulo operator because the actual string and the values are separated using modulo operator.

Let us execute the code and observe the output. Let us say 6, the output is displayed in the same format. One may get confused by looking at these four different print statements printing same results. But no need to worry as this is trivial part of the language, we are teaching you all these different types of print statements for the sake of completeness. Otherwise, even knowing single way of writing this print statement is sufficient enough any program.

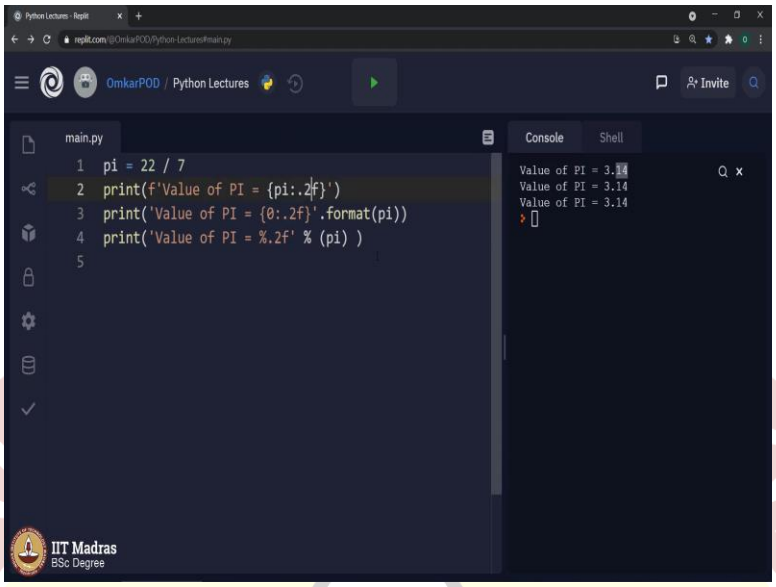
Let us look at one more python program using f print format function and string modulo operator. Then we will see why formatted printing is much more useful than this standard way of writing the print statement.

(Refer Slide Time: 13:35)



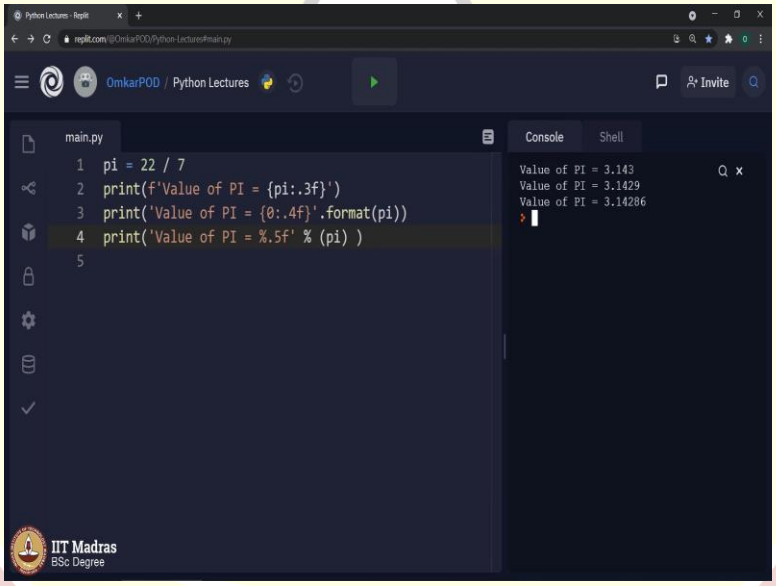
```python
pi = 22 / 7
print(f'Value of PI = {pi}')
print('Value of PI = {0}'.format(pi))
print('Value of PI = %f' % (pi) )
```

```
Value of PI = 3.142857142857143
Value of PI = 3.142857142857143
Value of PI = 3.142857
```



```python
pi = 22 / 7
print(f'Value of PI = {pi:.2f}')
print('Value of PI = {0:.2f}'.format(pi))
print('Value of PI = %.2f' % (pi) )
```

```
Value of PI = 3.14
Value of PI = 3.14
Value of PI = 3.14
>
```



```python
pi = 22 / 7
print(f'Value of PI = {pi:.3f}')
print('Value of PI = {0:.4f}'.format(pi))
print('Value of PI = %.5f' % (pi) )
```

```
Value of PI = 3.143
Value of PI = 3.1429
Value of PI = 3.14286
>
```

Look at this particular python program, pi is equal to 22 divided by 7, print, which is f print, value of PI is equal to pi. Next, using format function and third one using modulus operator. In this case, value of pi is going to be a fractional value because of this division operation. Hence, we will use percent f, which stands for float. Let us first execute this code and then, we will see how we can modify this formatted printing statements to get a different variation of the same output.

If you observe the output, for first two print statements, we are getting this particular value for pi. Whereas, for third statement, we are getting a value which is approximately equal to the first value. But over here, number of digits after the decimal point are only 6. This is happening because, as I mentioned, this is an old way of writing a print statement. Because of that, this particular string modulus operator has some limitations.

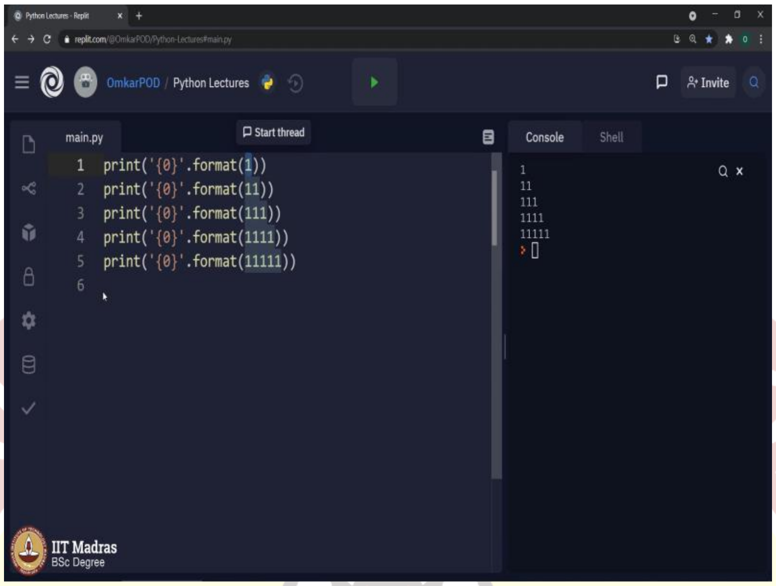
Irrespective of that, the concept which we want to discuss over here is related to this value of pi. Usually, while calculations, we use the value for pi as 3.14 just to simplify the calculations. Now, the question is, how to tell computer that I do not want this big number as a value of pi, I am happy with 3.14. Which means, how to tell the computer that I want only 2 digits in the fractional part of this particular value.

This can be done using something called as format specifiers. Which means, we can tell the computer what should be the format of the output value. Let us look at this modified code, pi colon .2f. Similarly, 0 colon point .2f. Once again, instead of percent f, percent .2f. First, let us execute this modified code. Then we will see, why we are getting such an output.

Now, you can see, we are getting 3.14 in all three instances. This is happening because we are explicitly telling the computer that I want the value of pi but this time, I want it in this particular format, .2f format. .2f stands for only 2 digits after decimal point. Because of this specification, computer will print only fraction up to 2 digits. We can make it 3, 4 or 5. Let us try this.

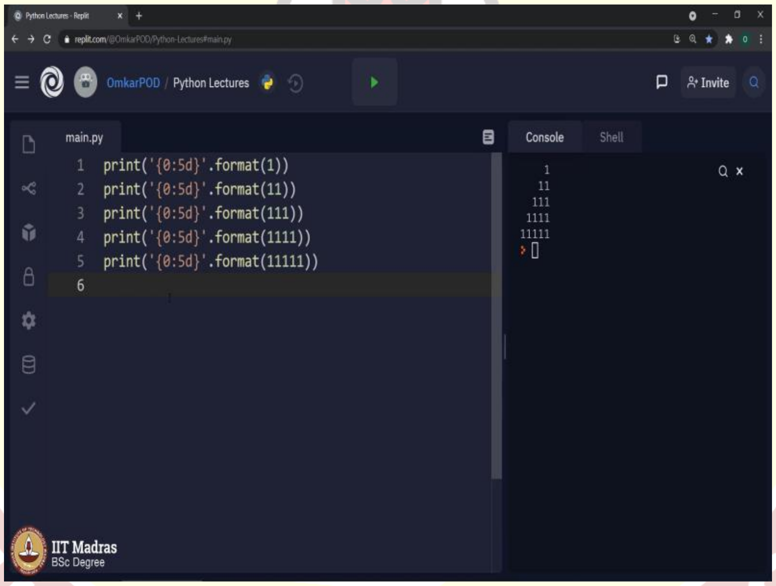
As expected, here we are getting 3 digits, then 4 digits and 5 digits. So, whatever the number we mention over here, those many number of digits will be provided after the decimal point. This same format specifier is very useful in order to display the output in a specific manner. Let us look at one more example of pattern printing which we have seen earlier.

(Refer Slide Time: 17:39)



```python
print('{0}'.format(1))
print('{0}'.format(11))
print('{0}'.format(111))
print('{0}'.format(1111))
print('{0}'.format(11111))
```

```
1
11
111
1111
11111
>
```



```python
print('{0:5d}'.format(1))
print('{0:5d}'.format(11))
print('{0:5d}'.format(111))
print('{0:5d}'.format(1111))
print('{0:5d}'.format(11111))
```

```
    1
   11
  111
 1111
11111
>
```

In this Python program, we are simply printing numbers 1, 11, 1 three times, 1 four times, 1 five times. But instead of printing it directly using the standard print statement, we are printing it using format function. If you remember, you have already executed this program to print this particular pattern. But now, I do not want this particular pattern to be printed which is left aligned, which means all the numbers are in line with each other on the left side.

Instead of that, now I want it right aligned. It is easily possible to do by adding some extra spaces with these values. But if we want to do it for a very long pattern, then it becomes a tedious task. In order to avoid that tedious process, we can use this format specifier to print the same pattern with right indentation. 0 colon, this is the place where we will mention and indicate the computer that what kind of a specific format we are looking for.

As you can see, the maximum length is 5 characters. Hence, we can say 5 d. This number 5 will indicate the computer that we want to use minimum 5 characters in print statement. And as we are using 1, 11, 111 as integers, that is the reason we are using d, because we have seen earlier, d stands for integer. Let us add this particular format specifier in each print statement. Let us execute and observe the output. As expected, we are getting the same pattern but this time, it is right aligned.

Before closing this lecture, let us summarize what all things we have seen in this particular lecture. We started with two parameters of print statement, end and sep which stands separate. Then, we saw three different types of formatted printing. First, using f print, second using format function and third using string modulo operator.

After that, we saw something called as format specifier, where we can explicitly tell the computer exactly in what format the output should be printer. Either, we can control the length of a particular output or we can also control number of digits after the decimal point. As I mentioned earlier, all these concepts are not that critical but they are very useful and they also makes our life, as in, the programmers life easier when we want to print the output in a certain way.

Thank you for watching this lecture. Happy learning.