INDIAN INSTITUTE OF TECHNOLOGY MADRAS

सिद्धिर्भवति कर्मजा

IIT Madras

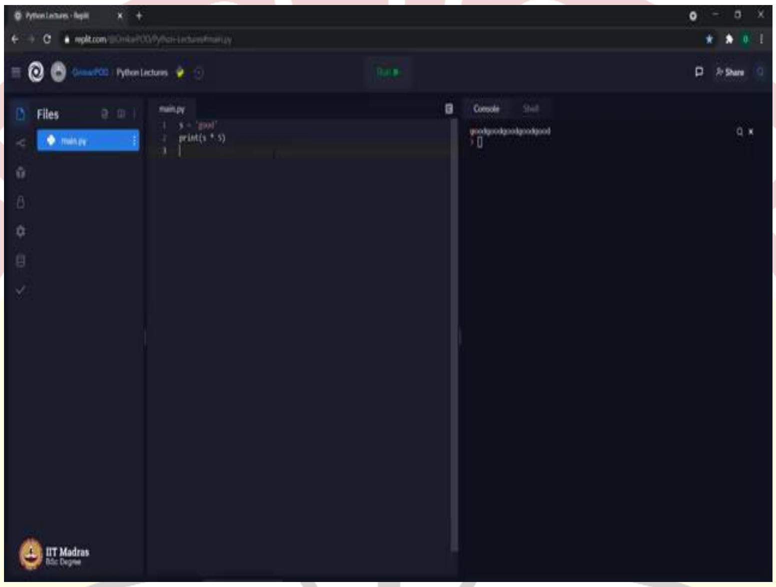
ONLINE DEGREE

Programming in Python

Professor Sudarshan Iyengar
Department of Computer Science and Engineering
Indian Institute of Technology Ropar
Omkar Joshi
Course Instructor
Indian Institute of Technology Madras Online Degree Program

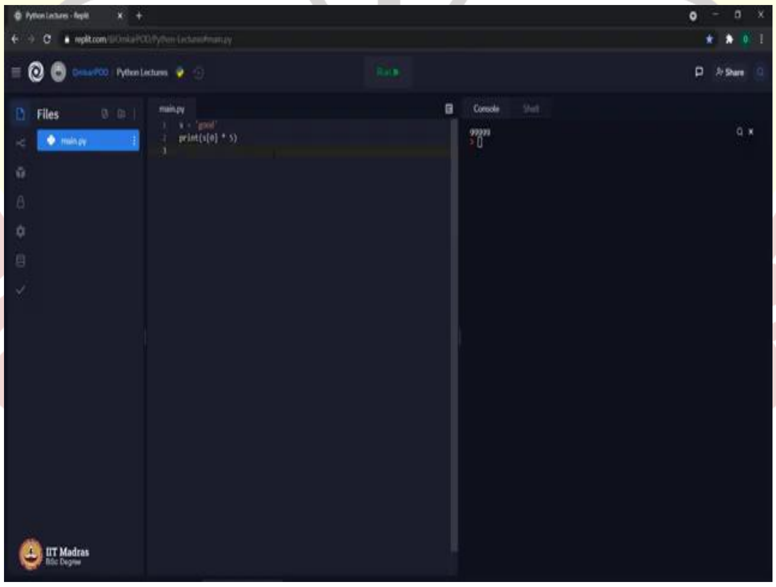
More on Strings

(Refer Slide Time: 0:16)



```
s = 'good'
print(s * 5)
```

```
goodgoodgoodgoodgood
> 
```



```
s = 'good'
print(s[0] * 5)
```

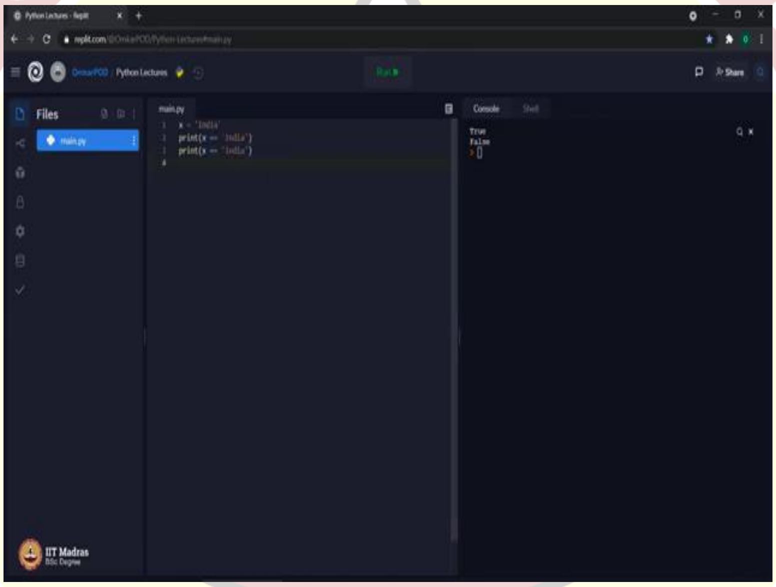
```
ggggg
> 
```

Hello Python students. In last lecture we saw various concepts related to strings, like concatenation, indexing, and slicing. In this lecture we will see few more concepts related to strings. In the first concept, it is called replication. Let us declare one variable s as a string, value is good, print, so far we have seen what happens when we use a plus operator or a minus operator with string.

Now, we will see what happens if we use multiplication operator with string. s star 5. So, what you think? What should be the output of this particular Python program? Will it encounter an error or will it print something? Let us try it. It prints the string 5 times one after the another, which means it is like an concatenation operator where same string is concatenated with itself 5 times.

Let us take it one step further, instead of s if we make it s of 0, now what will be the output? Let us see, 5 times the letter g because s of 0 is g. This particular feature of string data type is called as replication where a string or a character from a string replicates itself based on the number with which it is multiplied by.

(Refer Slide Time: 2:04)



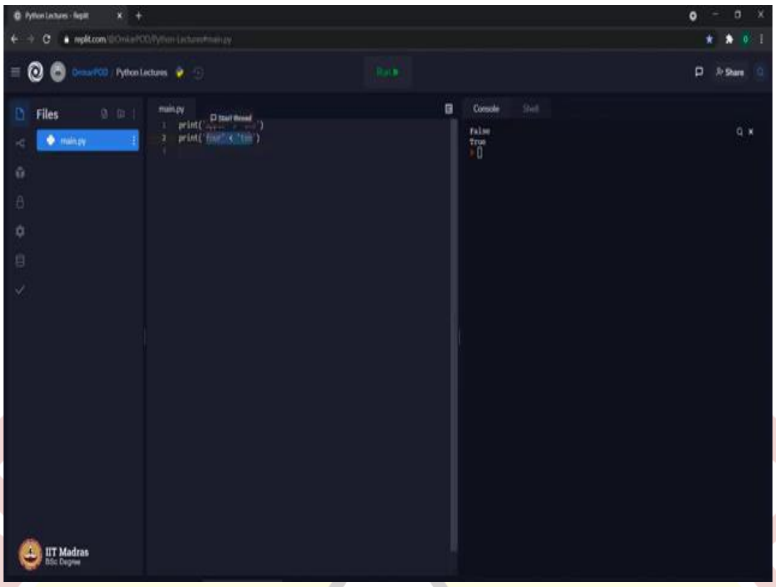
```
x = 'India'
print(x == 'India')
print(x == 'india')
```

```
True
False
> 
```

Now the next string property is string comparison. Let us write one small piece of code. x is equal to India, print x equal equal to India and print x equal equal to India but over here this ‘i’ is small. So the question is what will be the output of this kind of comparison. A letter being capital or small, does that make any difference or still it will consider the string India as same which is like variable x.

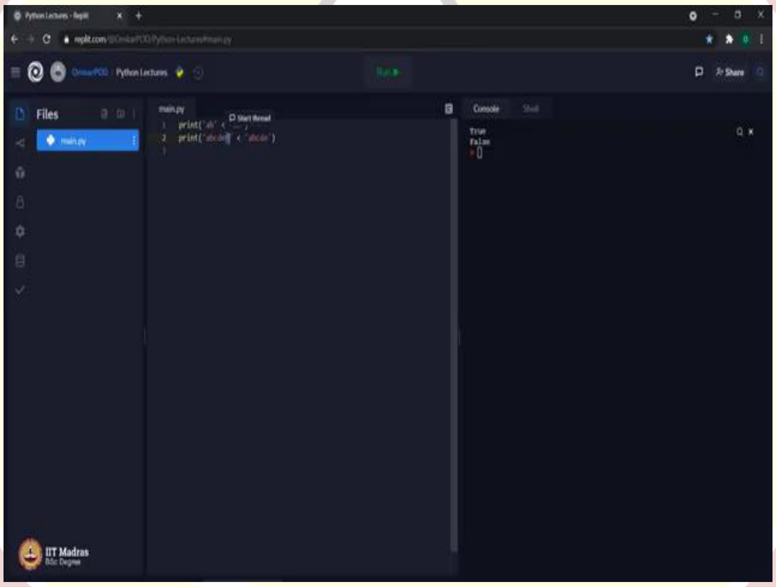
Let us execute the code, first is true, obviously because we have ‘I' as capital, but in second case it is false, which means a single letter change from capital to small letter is considered as not equal, therefore x equal equal to India with small ‘i’ is resulting into false. Let us see some more examples where string comparison is happening.

(Refer Slide Time: 3:17)



```
print([illegible])
print('four' < 'ten')
```

```
False
True
```



```
print('ab' < [illegible])
print('abcde' < 'abcde')
```

```
True
False
```

Print apple greater than 1 and print four less than 10. What you think, what will be the output of such string comparison? First, even is it possible to compare strings like this or will it result into an error? Let us figure out there is no error. It shows some output false and true, now let us see why false and true. First when we compare apple greater than one, it says false.

If you see a string apple is longer or has more characters than the string one, still it is not greater. Similarly, in second case when we compare 4 less than 10, it says true. But we know 4 is not less than 10. But that is true when we compare 4 and 10 as integers, here the word 4 and 10 are strings. Hence, the comparison operator works in a different manner.

Let us see how these comparison operators greater than and equal to works with strings. A computer starts the comparison process character by character as in the zeroth letter of first string which is a in this case will be compared against the zeroth letter of second string, which is o in this case. That means a computer will compare whether a is greater than o.

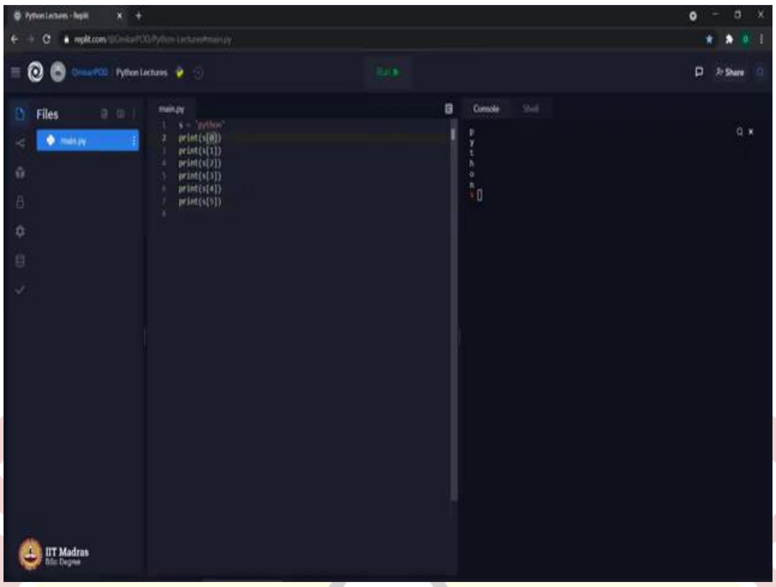
As we know in alphabetical order a comes before o, which means a is not greater than o, because of that it results into false. Similarly, in second case the letter f is less than t because the letter f comes before letter t in an alphabetical order, because of that it says true. Now, you must be thinking what happens if first letters are equal.

Let us try to have one example where the first letter is equal and see what happens next. Print ab less than az. In this case the first letter a is equal, but the second letter b and z and are different. Similarly, in second print statement first couple of letters are same, in fact, all the letters from second string are exactly identical with initial letters from first string, except the last one character.

Let us see what output this particular string comparison gives us? It says true followed by false, which means ab is less than az. This is happening because when the first character is equal computer moves to the next character which is b and z. As b comes before z in alphabetical order it says true. Now, in case of second print statement the first letter as in the zeroth letter of first string which is a, which is matching to the zeroth letter of second string.

Same, the second letter which is b, next letter c, next letter d, and then next letter e. All these letters are matching. In first case after e there is one letter which is f, but on the right hand side of that letter than operator there is no letter to compare after e because of that computer end up comparing f with nothing and due to that the output is false because f cannot be less than nothing.

(Refer Slide Time: 7:51)



```
s = 'python'
print(s[0])
print(s[1])
print(s[2])
print(s[3])
print(s[4])
print(s[5])
```

```
p
y
t
h
o
n
```

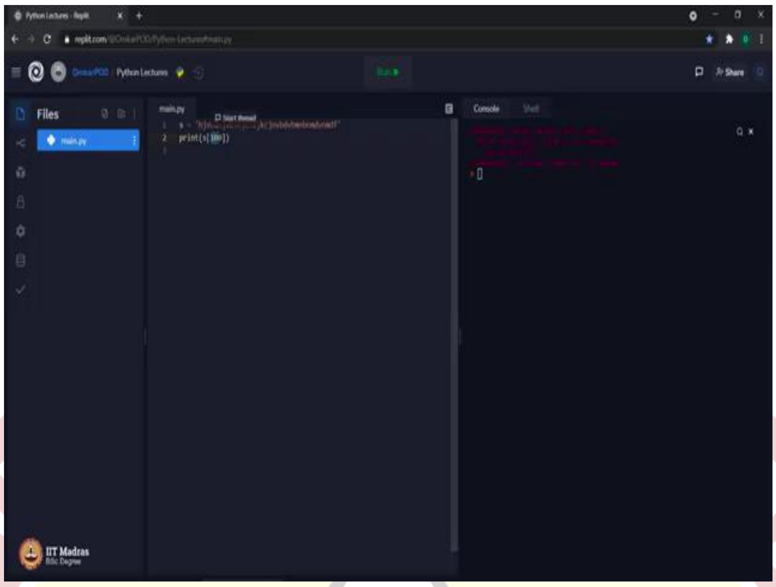
Next string concept is string indexing. But in this lecture we will see different kind of string indexing called as negative indexing. Let us declare one string. s is equal to Python, print s of 0, s of 1, s of 2 and so on and the output is t, y, t, h, o, n as expected. This is something we have already done. What if we do something different this time?

Instead of giving these indexes 0, 1, 2 and so on can we give negative numbers? For example, minus 1, minus 2, minus 3, minus 4, minus 5, and minus 6, what do you think, what will be the output of this particular Python program? Can we even access minus 1 letter in this particular string? We already know p is zeroth letter, y is first, t is second and so on, n is the fifth letter. Is it even possible to access negative index for a string?

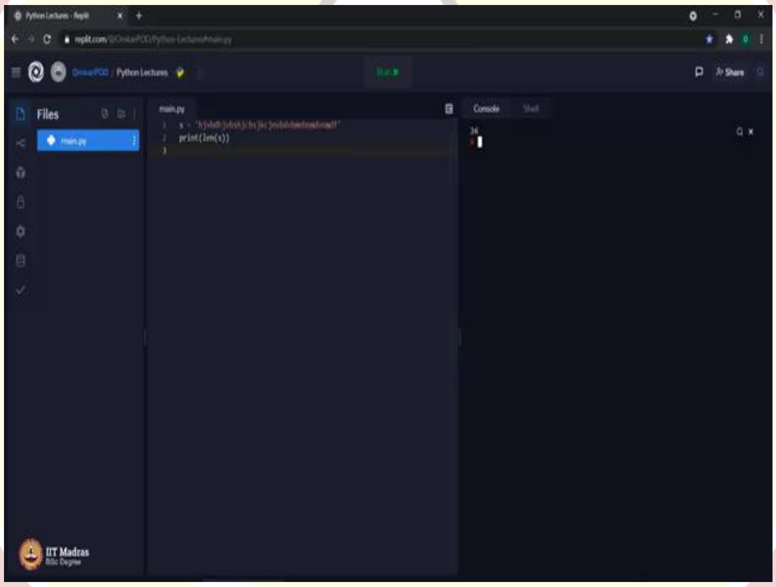
Let us execute and check. Yes, it is possible. If you see we are not getting any error. In fact, we are getting the exact letters from the string but this time they are in the reverse order, minus 1 that is showing n, for minus 2 it is showing o, for minus 3 it is h, for minus 4 it is t, and so on, for minus 6 it is showing p, which is the first letter of the string.

Which means whenever we do minus 1, it access the last letter of the string, minus 2 is second last letter, minus 3 is third letter from the end, minus 4 is fourth letter from the end and so on. This particular indexing is called as negative indexing.

(Refer Slide Time: 10:19)

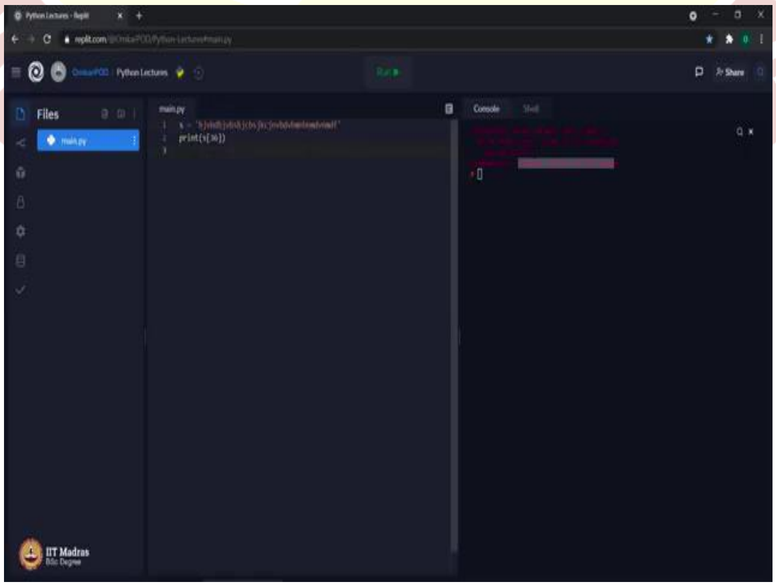


```
s = '[illegible]'
print(s[100])
```

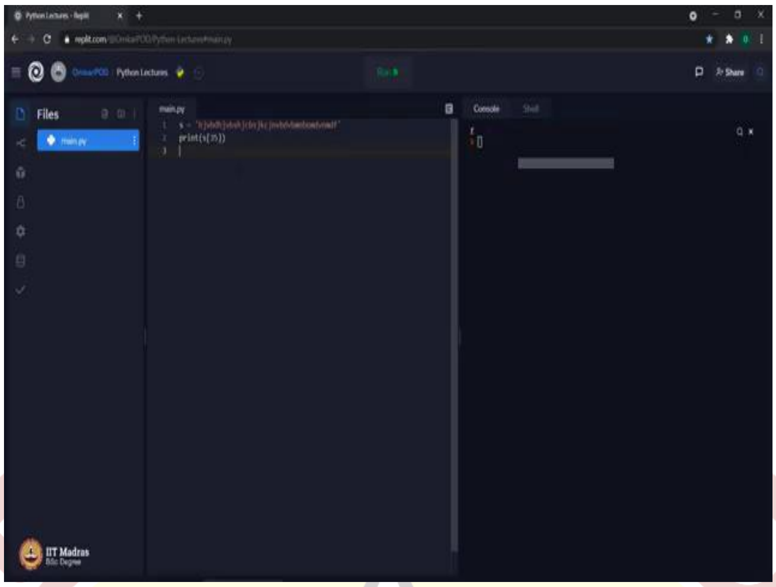


```
s = '[illegible]'
print(len(s))
```

```
36
```



```
s = '[illegible]'
print(s[36])
```



```
s = '[illegible]'
print(s[35])
```

```
f
```

Let us see the next concept related to string. First let us declare one string variable and let us have some very big value again this variable, some random character. Let us try to print s of let us say 100, we do not know whether we have 100 letters in the string or not. The string looks big enough but may not be big enough to have 100 characters. Let us execute and see what happens?

It shows an error which says 'String index out of range', which means we are trying to access the 100th character in the particular string but as per the error it says index out of range, which means the 100 character does not even exist in the given stream, due to which computer cannot print it. Now, the question is if we have such a big string, maybe even bigger like this, still the output is index out of range.

In case of such a big string, how to figure out exactly how many number of characters are there in that string so that we will not end up using some index value which is out of range. For that there is one command called as len, which means length, print of length of s. Let us execute and the output is 36.

This number 36 represents the total number of characters in this particular string. Now as we know there are 36 characters, which means now we should be able to execute s of 36. What you think will this code execute or still we will get error? Let us see, still we are getting the same error, string index out of range, but we already calculated that the length is 36, still it is giving error.

Can you tell me why this particular thing is happening? Because if you remember in Python the string indexing starts from 0, due to which whenever we say there are 36 letters that means the indexes are from 0 to 35, let us make it 35, and see whether we are getting the last character of the string or not?

Yes, indeed we are getting the letter f, which is the last letter in that particular string which is the 35th character, which indicates that we can use the index number which is 1 less than the total length of that particular string. In this case the length of string is 36, therefore the highest possible index is 35. Thank you for watching this lecture. Happy learning!