

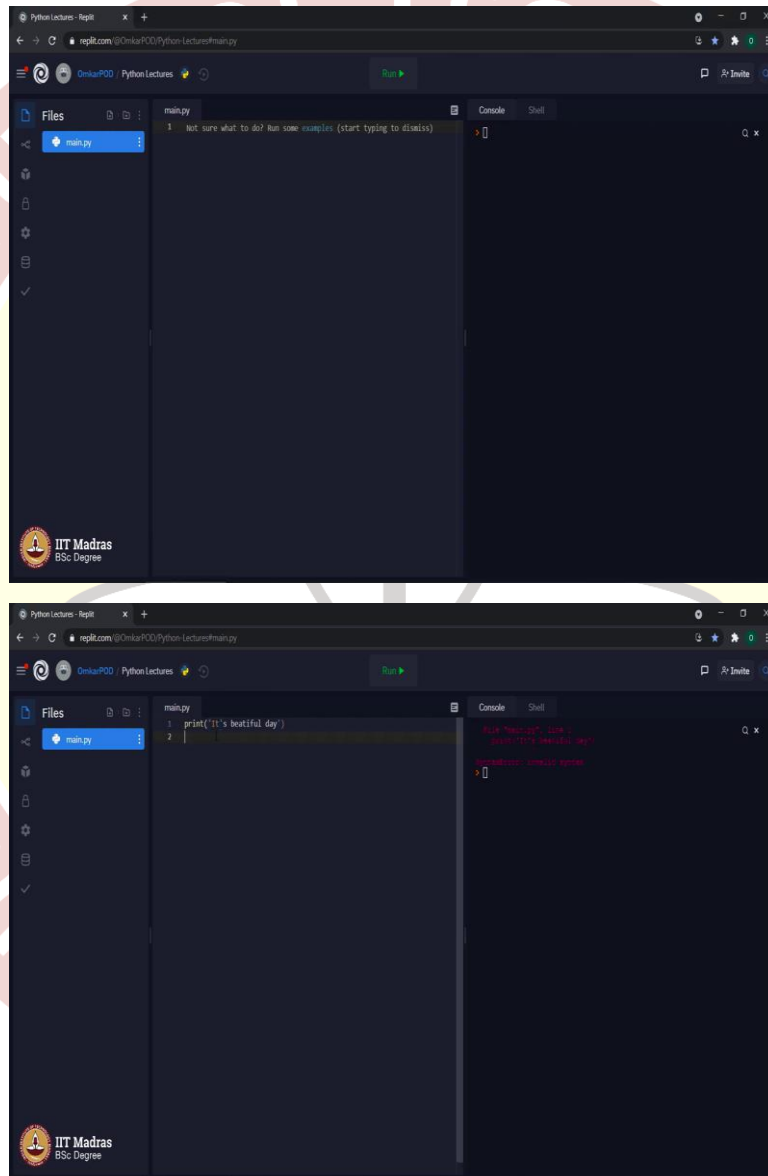


IIT Madras

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
Escape Characters and Types of Quotes

(Refer Slide Time: 00:16)

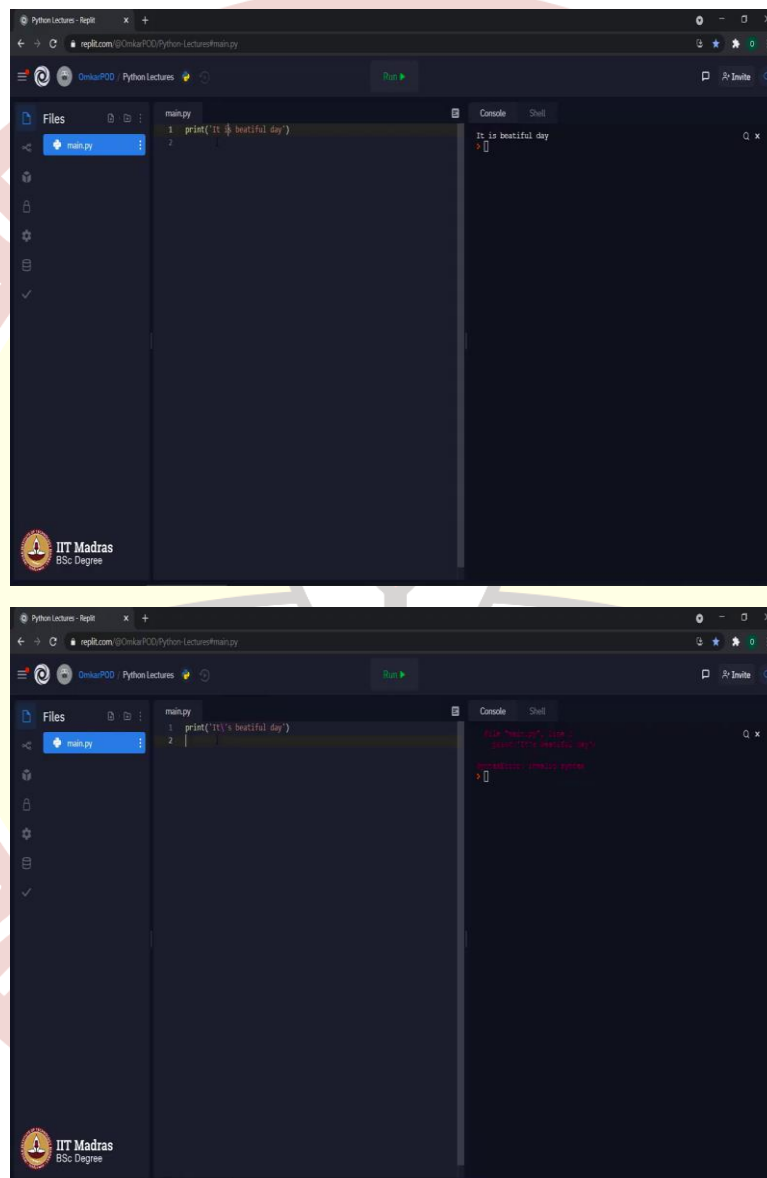


The image displays two screenshots of a Python REPL environment, likely from a video lecture. The top screenshot shows a file named 'main.py' with a single line of code: `1 not sure what to do? Run some examples (start typing to dismiss)`. The bottom screenshot shows the same file with two lines of code: `1 print('It's beautiful day')` and `2` on the next line. The console output on the right shows the execution of the first line, resulting in `It's beautiful day`. The background features a large, semi-transparent watermark of the Indian Institute of Technology Madras logo.

Hello Python students. In this lecture, we will see two new concepts, namely escape characters and use of quotes supported by the Python language. Let us start with escape characters. For example, I want to print a statement like, It's a beautiful day. As you can see, I have started with a single quote and I have end this particular string with a single quote, which is a valid syntax for a print statement.

Let us execute. It says invalid syntax, because over here, It's has this apostrophe and computer does not differentiate between this apostrophe and the single quote, which we usually use to start a string or stop a string, because of that, a computer starts a string with this quote and ends it with this particular quote which is actually the part of that sentence. Now the question is how to solve this problem.

(Refer Slide Time: 1:40)



The image contains two screenshots of a Python REPL environment, likely Replit, showing a syntax error and its resolution. The top screenshot shows a file named `main.py` with the following code:

```
1 print('It's beautiful day')
2
```

The console output shows the error message:

```
It is beautiful day
>
SyntaxError: invalid syntax
```

The bottom screenshot shows the same file with the code modified to use double quotes:

```
1 print("It's beautiful day")
2
```

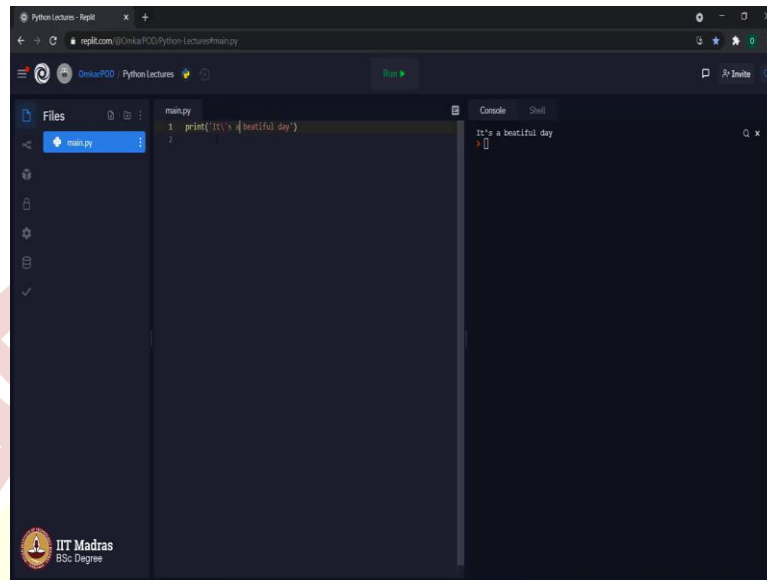
The console output now shows the correct output:

```
It's beautiful day
>
```

One easiest solution is to make it, it is and it will work. But I do not want to do that, I want to keep it like this and this particular problem or this particular error can be fixed using escape character. Escape character is a mechanism which allows us to add those specific characters or those specific symbols in a string in print, which usually we cannot do, as you can see over

here, with this single quote. Escape character is written using a backslash followed by the character, which you want to insert in that particular string.

(Refer Slide Time: 2:34)



The screenshot shows a web-based Python REPL interface. The file explorer on the left shows a file named 'main.py'. The code editor in the center contains the following Python code:

```
1 print('It's a beautiful day')
2
```

The console output on the right shows the result of the execution:

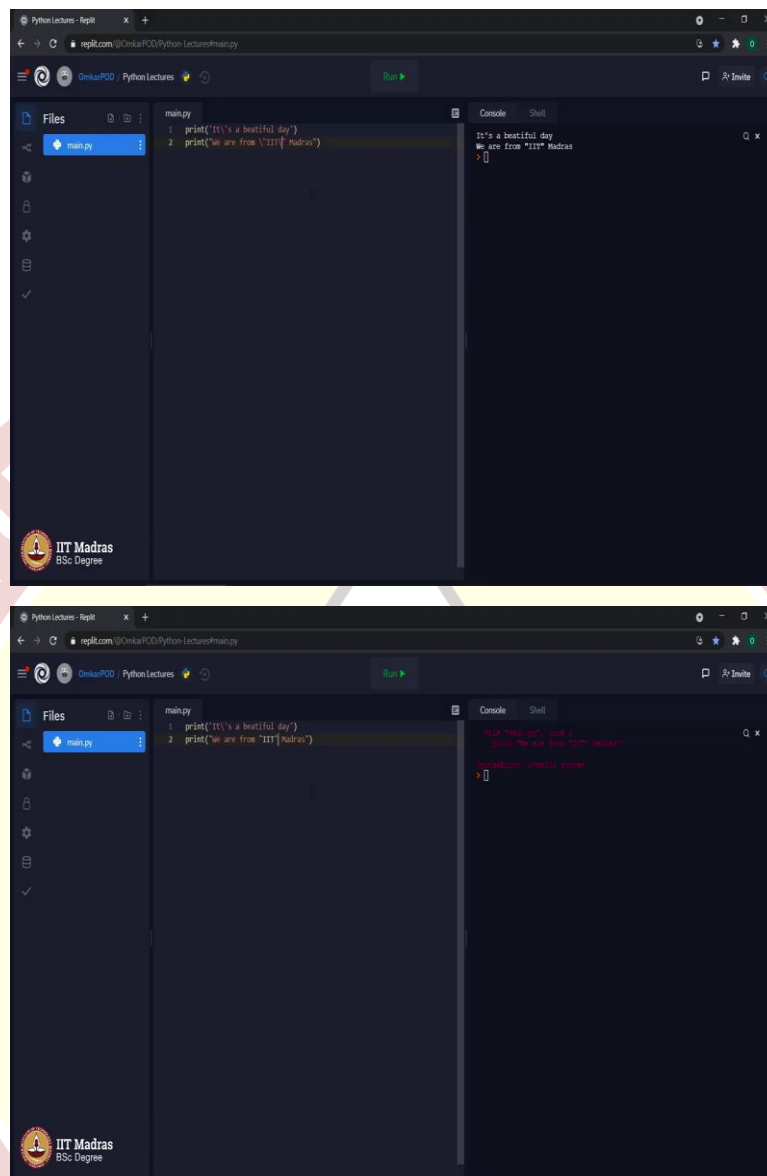
```
It's a beautiful day
```

The background of the slide features a large, faint watermark of the IIT Madras logo, which includes a lamp and the text 'IIT Madras BSc Degree' and 'सिद्धिर्भवति कर्मजा'.

In this case, I want to insert a single quote. Hence, I should use backslash before that single quote. As you can see, as soon as I insert a backslash, even the color of that particular statements changes immediately. Now we can observe this backslash followed by a single quote has a different color, because now, these two characters are not considered as separate characters.

Computer will translate this particular character as a single entity and it will replace it with a single quote as we want it, just like this. It's beautiful day, it's a beautiful day and this particular character is referred as escape character.

(Refer Slide Time: 3:37)



The image contains two screenshots of a Python REPL interface, likely Replit, showing a file named `main.py` and its execution output in the console.

The top screenshot shows the code in `main.py`:

```
1 print('It's a beautiful day')
2 print("We are from 'IIT' Madras")
```

The console output shows the execution of the code:

```
It's a beautiful day
We are from "IIT" Madras
```

The bottom screenshot shows the same code in `main.py`:

```
1 print('It's a beautiful day')
2 print("We are from 'IIT' Madras")
```

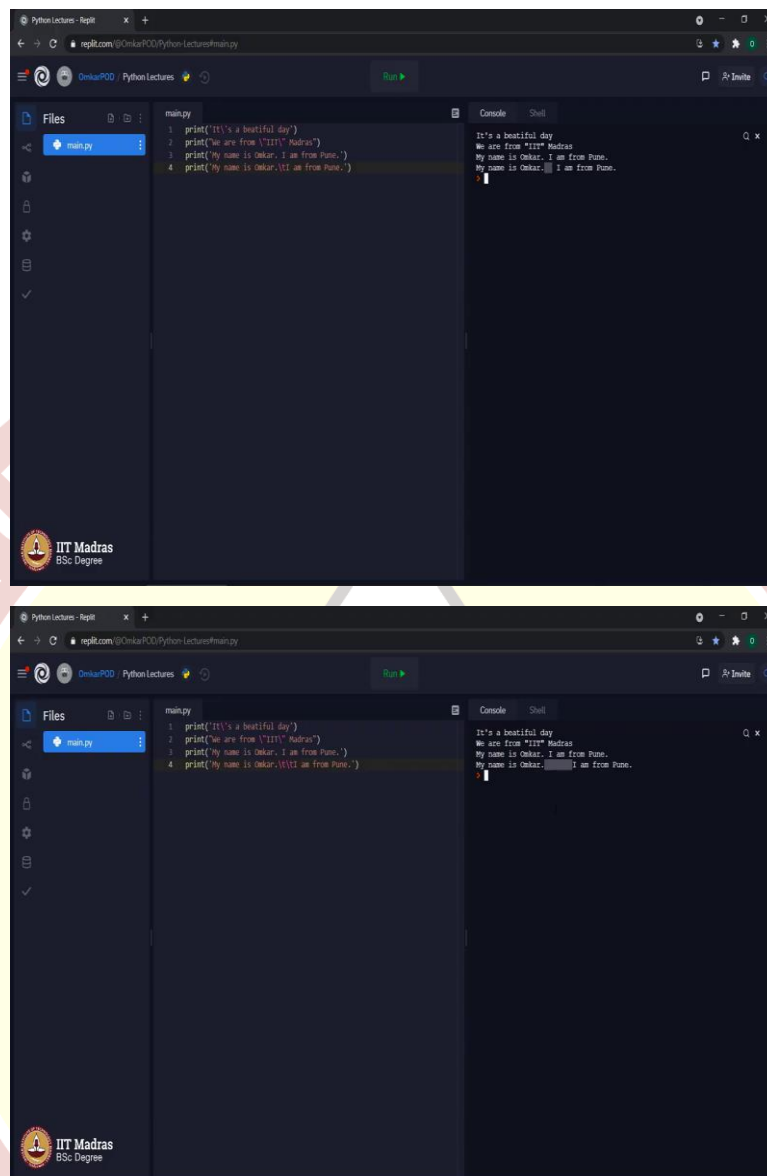
The console output shows an error message:

```
File "main.py", line 2
    print("We are from 'IIT' Madras")
                          ^
SyntaxError: invalid syntax
```

Let us try one more example, print. Now, instead of single quotes, I want to use double quotes as we have seen earlier, with respect to print, it is absolutely okay to use double quotes instead of single quotes, we are from IIT Madras, once again computer rates it is starting of the string and this double quote as ending of the string, then it is not able to process this particular term IIT in between followed by starting of a string and ending of a string.

Because of this computer is giving us an error, which says invalid syntax, this escape character will rescue us from this particular invalid syntax error, backslash followed by double quote, once again, backslash followed by double quotes. Now we are getting the expected output, we are from in double quotes, IIT Madras.

(Refer Slide Time: 04:51)



The image displays two screenshots of a Python REPL (REPL) interface, likely JupyterLab, showing the execution of a Python script. The interface includes a file explorer on the left, a code editor in the center, and a console on the right.

Top Screenshot: The code editor shows a file named `main.py` with the following code:

```
1 print('It's a beautiful day')
2 print("We are from 'IIT' Madras")
3 print("My name is Omkar. I am from Pune.")
4 print("My name is Omkar. IIT am from Pune.")
```

The console shows the output of the code:

```
It's a beautiful day
We are from "IIT" Madras
My name is Omkar. I am from Pune.
My name is Omkar. IIT am from Pune.
```

Bottom Screenshot: The code editor shows the same code as the top screenshot. The console shows the output of the code, with the last two lines of the output being:

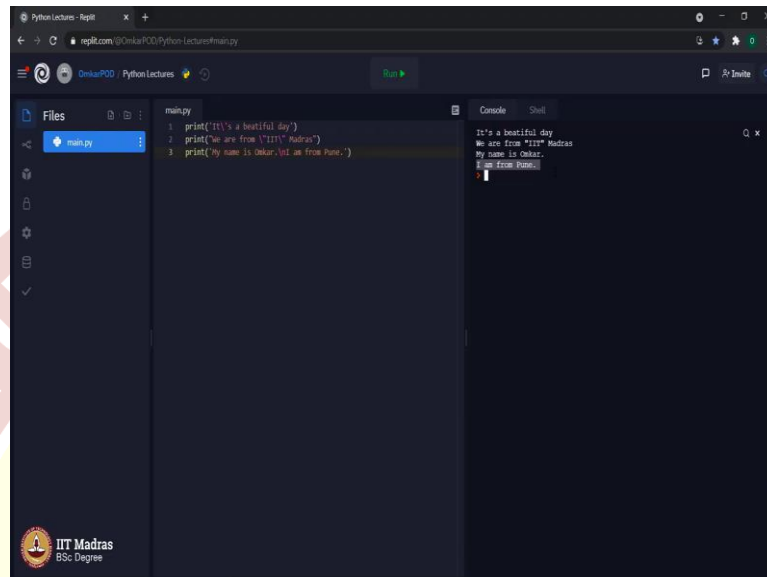
```
My name is Omkar. I am from Pune.
My name is Omkar. IIT am from Pune.
```

This is related to single quotes or double quotes. What if I want to do something a little bit different than this? For example, print My name is Omkar space, I am from Pune. If you print it, it will print correctly. But what if I want to give additional space between these two sentences? Currently, there is only one space in between these two sentences. But I want to separate these sentences even further.

The simplest way, could be to add more spaces, but it is not a good practice. So what to do, usually we use tab to give these extra spaces. But how to use that in the Python language? Let us retain this quote and copy it. Instead of this space if I use something like backslash t, t stands for tab, if you can observe the last two lines of the output here, there was only single space.

But now you can see more gap between these two sentences. If we add one more t over here, it will put more gap between these two sentences and now we do not even have to give multiple spaces between two sentences. So this particular escape character is called tab.

(Refer Slide Time: 6:29)



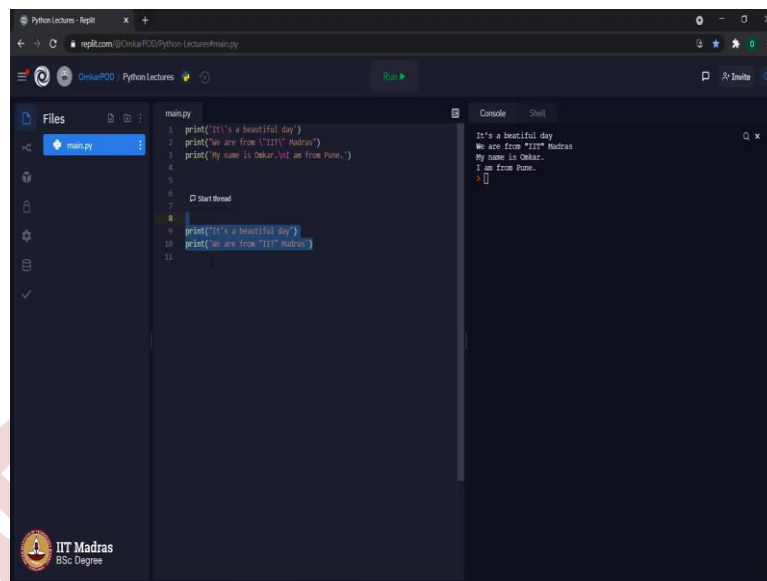
```
main.py
1 print("It's a beautiful day")
2 print("We are from \"IIT\" Madras")
3 print("My name is Onkar, but we are from Pune.")
```

```
Console
It's a beautiful day
We are from "IIT" Madras
My name is Onkar,
but we are from Pune.
```

Now there is a sufficient gap between these sentences. But still, I am not happy with this particular quote. I want to print these sentences on the different lines. A straightforward way is to remove this particular sentence from here and add it as a fourth print statement. But we can avoid that by placing one more escape character called backslash n. n stands for new line. Let us execute this quote.

Now you will see the first sentence is getting printed on the third line whereas next sentence is getting printed on the fourth line in the output even though these two sentences are actually written as a part of same print statement. This is happening because of this specific escape character called backslash n, which refers to a new line.

(Refer Slide Time: 07:24)



The screenshot shows a Python REPL window with a file named `main.py`. The code in the file is:

```
1 print('It's a beautiful day')
2 print("We are from 'IIT' Madras")
3 print("My name is Oskar, I'm from Pune.")
4
5
6
7
8
9 print('It's a beautiful day')
10 print("We are from 'IIT' Madras")
11
```

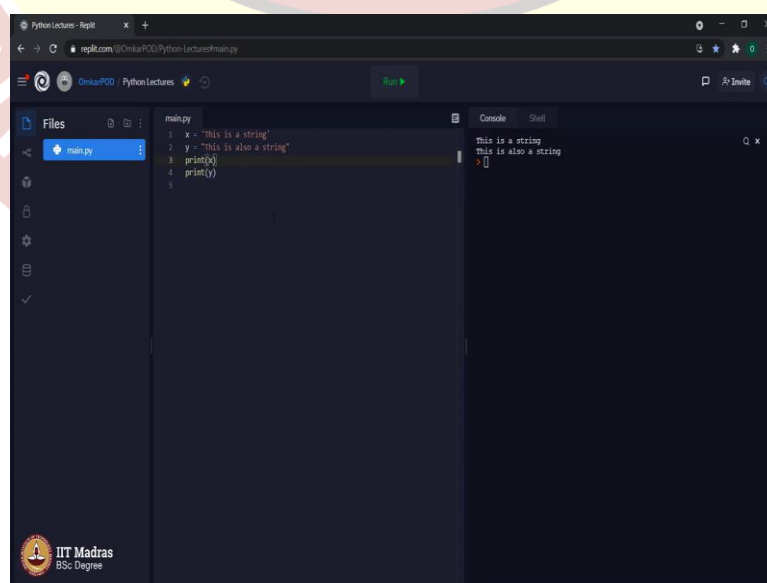
The console output is:

```
It's a beautiful day
We are from "IIT" Madras
My name is Oskar,
I'm from Pune.
```

Before moving to the next concept let me give you a small exercise, let us copy these two lines, I will change the single quotes to double quotes and remove this escape characters involved. Similarly, I will make these single quotes and remove this particular escape character. Oh, there is a spelling mistake here. That is not important part.

The exercise is, I will not execute these two lines, it is up to you to execute these two lines, see the output and figure out why the specific output is coming like that. This is very important concept of strings. But as it is very simple, I am leaving it up to you guys to study on your own.

(Refer Slide Time: 08:29)

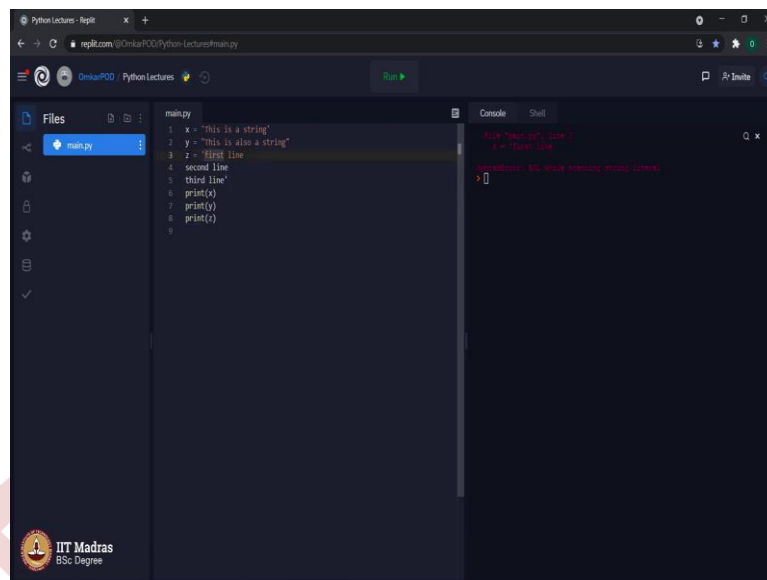


The screenshot shows a Python REPL window with a file named `main.py`. The code in the file is:

```
1 x = "this is a string"
2 y = 'this is also a string'
3 print(x)
4 print(y)
5
```

The console output is:

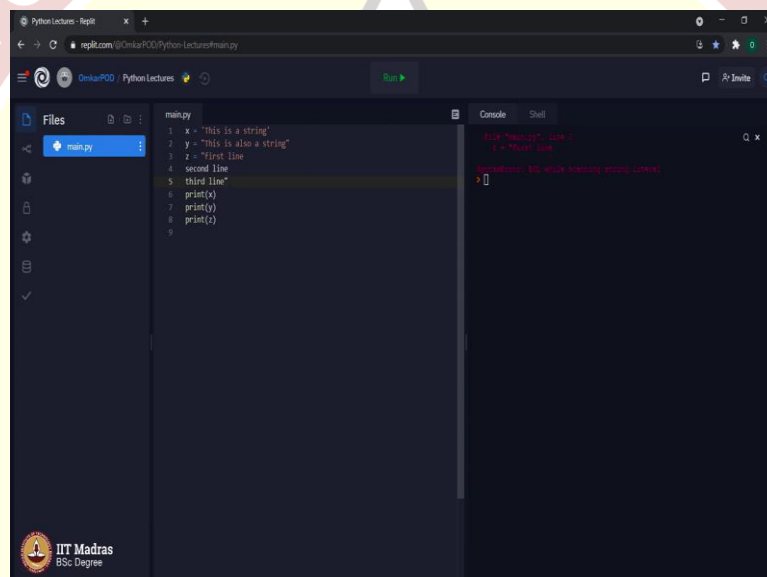
```
this is a string
this is also a string
```

```
main.py
1 x = "this is a string"
2 y = "this is also a string"
3 z = "first line"
4 second line
5 third line"
6 print(x)
7 print(y)
8 print(z)
9

Run

Console
Print "main.py", line 1
x = "first line"
SyntaxError: EOL while scanning string literal
>
```



```
main.py
1 x = "this is a string"
2 y = "this is also a string"
3 z = "first line"
4 second line
5 third line"
6 print(x)
7 print(y)
8 print(z)
9

Run

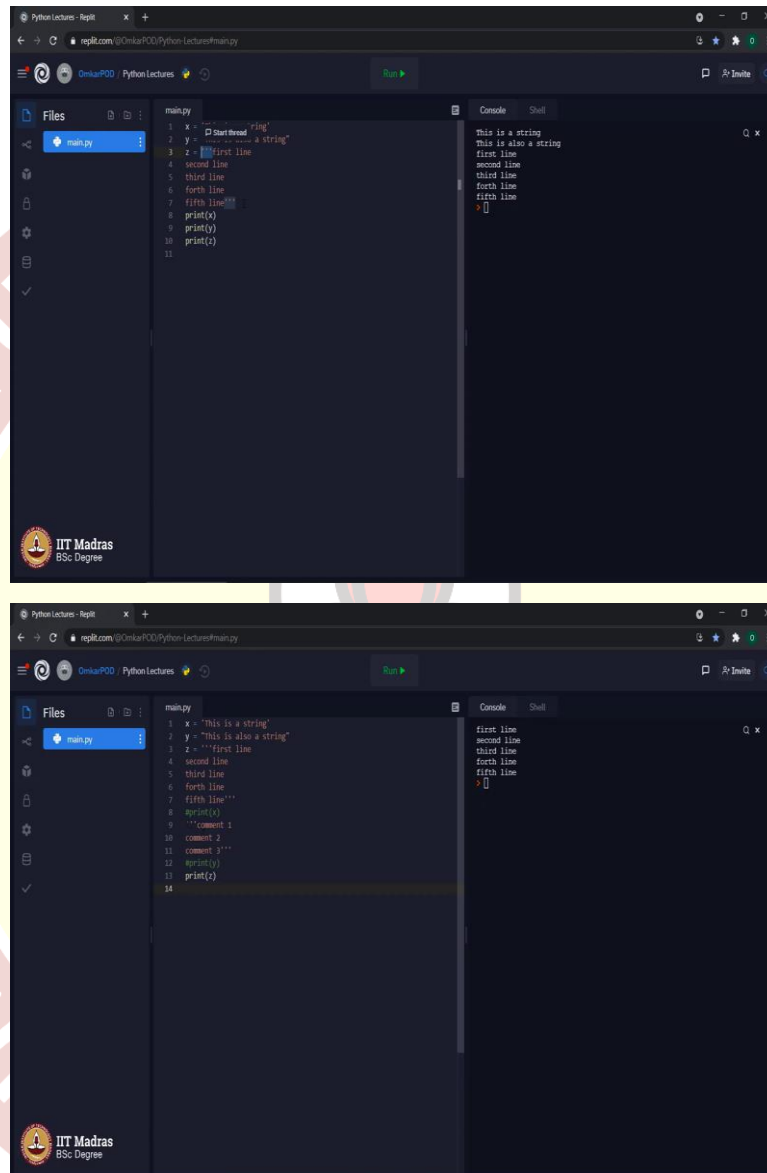
Console
Print "main.py", line 1
x = "first line"
SyntaxError: EOL while scanning string literal
>
```

Next part of this particular lecture is use of quotes supported by the Python language. We have already seen the use of single quote and double quote for strings; x is equal to single quote this is a string, y is equal to double quotes, this is also a string, print x print y. As expected, it will print these two strings. So far it was working. But for example, I want to write a third string which says; first line, second line, third line print z. It says EOL while scanning string literal.

EOL stands for end of the line, it says end of the line while scanning string literal. When computer starts scanning this third line; starts with z is equal to single code first line. It is still looking for the closing of this single quote which it is not able to find as part of this third line. That is why we are getting this particular error which means this single quote is useful only when a string is stored in a single line.

If we divide the string in multiple lines, then we cannot use single quotes. Let us try double quotes instead of that, still we are getting the same error. Now, the question is how to handle this kind of string?

(Refer Slide Time: 10:19)



The image contains two screenshots of a Python IDE interface, likely Replit, showing code for multiline strings. The top screenshot shows a file named 'main.py' with the following code:

```
1 x = "This is a string"
2 y = "This is also a string"
3 z = "first line
4 second line
5 third line
6 forth line
7 fifth line"
8 print(x)
9 print(y)
10 print(z)
```

The console output shows:

```
This is a string
This is also a string
first line
second line
third line
forth line
fifth line
```

The bottom screenshot shows the same file 'main.py' with the following code:

```
1 x = "This is a string"
2 y = "This is also a string"
3 z = """first line
4 second line
5 third line
6 forth line
7 fifth line"""
8 print(x)
9 """comment 1
10 comment 2
11 comment 3"""
12 print(y)
13 print(z)
14
```

The console output shows:

```
first line
second line
third line
forth line
fifth line
```

In order to store such a string, which is distributed over multiple lines is stored using three quotes. Let us print and try as you can see, it says first line, second line, third line. Similarly, we can keep adding as many lines as we want, fourth line, fifth line and so on. So, now, we know there are three different types of quotes, which are supported by Python language; single quote, double quote and then triple quotes, which we studied just now, and it is used for storing multiline strings.

There is one more advantage is using this triple quotes. Earlier we have studied if we use hash, then computer ignores that particular line from execution, it did not print x and y, it printed only the variable z, but ignored variable x and y which we printed in line number 8 and 9 because the use of this particular character hash.

As we know hash is used to comment a particular quote, which means that particular line will be considered as a comment, not a part of Python program. But what if I want to give a comment, which occupies more than one line, I can always say, comment 1, comment 2, comment 3 and so on and it should work.

But adding this hash every time is not optimum solution. The similar problem can be solved once again using these triple quotes. Because once again, whenever we use triple quotes, it becomes a comment in Python. So today, we have studied a new type of quotes called triple quotes and they can be used for two different purposes. First; to define a multiline string and second to give a multiline comment. Thank you for watching this lecture. Happy learning!

