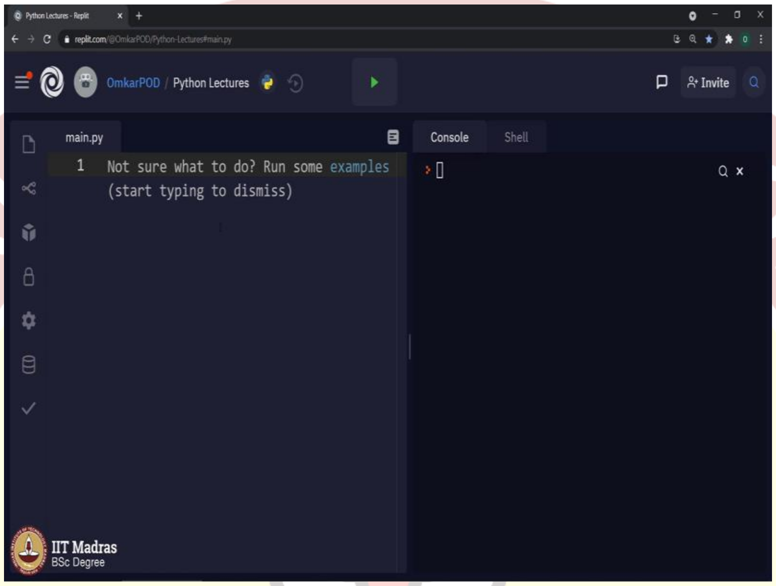
IIT Madras

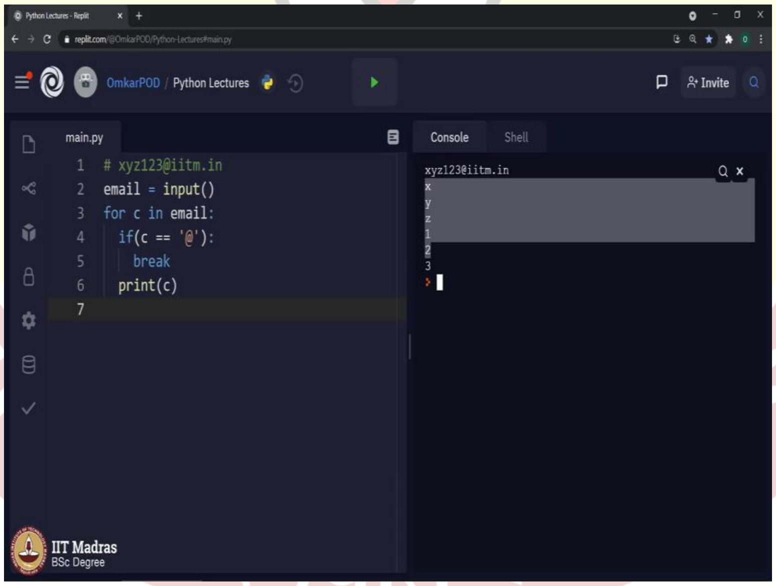
ONLINE DEGREE

Programming in Python
Professor. Sudarshan Iyengar
Department of Computer Science and Engineering
Indian Institute of Technology, Ropar
break, continue and pass

(Refer Slide Time: 00:16)

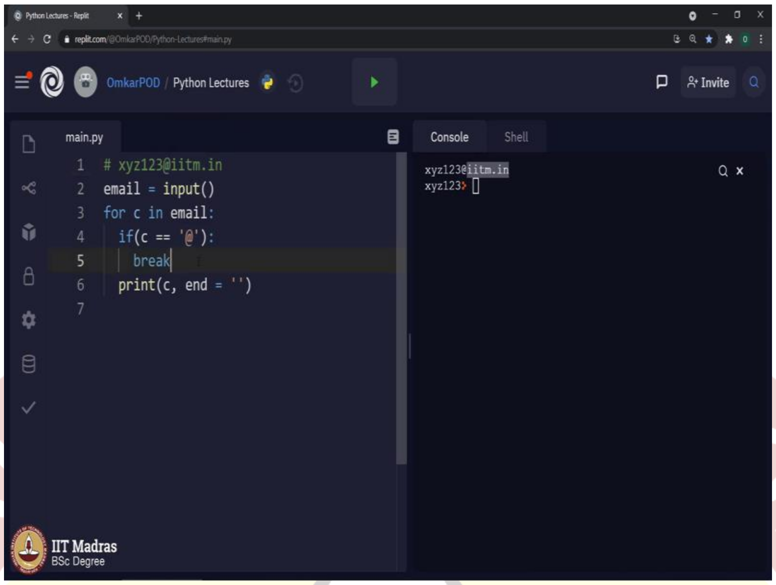


```
Not sure what to do? Run some examples
(start typing to dismiss)
```



```python
# xyz123@iitm.in
email = input()
for c in email:
  if(c == '@'):
    break
  print(c)
```

```
xyz123@iitm.in
x
y
z
1
2
3
>
```



```python
# xyz123@iitm.in
email = input()
for c in email:
  if(c == '@'):
    break
  print(c, end = '')
```

```
xyz123@iitm.in
xyz123>
```

Hello Python students. In this lecture we will discuss 3 new keywords, break, continue, and pass, let us start with break. In order to demonstrate the use of this keyword break let us define a problem statement. I want to accept an email id from the user using the input statement and after that I want to print only the username part of that particular email id.

For example, my email id is xyz123 at the rate iitm.in. In this case the output should be only the username part of this particular email id which is xyz123, which means I have to write a loop in such a way that it will print every character in the input email id till it reaches this at the rate character, as soon as it reaches this character the loop should stop. Let us try to translate this into a Python code.

Email is equal to input for c as in character in email, if character is equal equal to at the rate, then we will use the keyword break or else we will print the character. First let us execute the code and then we will see how it is working. Input xyz123 at the rate iitm.in it is printing the letters in user name xyz123 as expected.

In this case what happens is a computer reads the value stored in variable email id which is in this case xyz123 at the rate iitm.in. So, the computer will start from x till n one character at a time. Hence the first value for this variable c will be x, then that x will be compared against at the rate, as they are not matching this particular if block will not execute and the next statement which is print c will execute.

Similar thing will happen for y, then z, 1, 2 and then 3. Then there will be a point where the value in variable c will hold this particular character which is at the rate sign, at that point c is equal equal to at the rate condition becomes true and this particular if block will be executed.

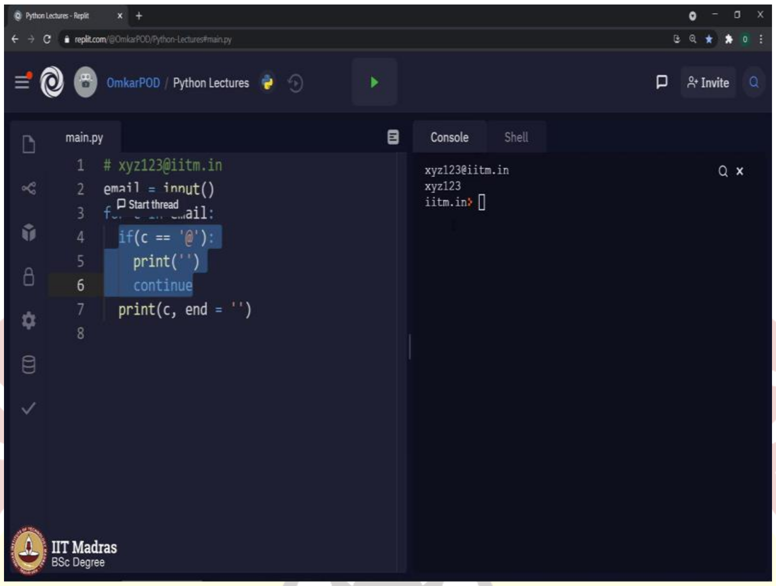
In this case this particular statement break will execute and the execution of this for loop will be stopped immediately. As soon as that happens computer will directly jump to the next line which is outside this particular loop which is this line number 7 and at line number 7 we have not written any code, hence, it will terminate the entire Python code.

So, based on this information we can easily relate this keyword break to a specific statement which we have studied earlier in computational thinking called exit loop. The functionality of exit loop and break is identical, it simply stops the execution of the current executing loop. And as I mentioned loop which means this break statement can be used inside for loop, a while loop or even inside nested loops.

Let us move to next keyword which is continue. Before starting with continue let me revise the problem statement. I want the username part of the email id in first line and I want the domain name of the email id in the second line which means the first line of the output should print xyz123 and the second line of the output should print iitm.in, as we have studied we can print these all letters in a single line by simply adding end is equal to empty quotes.

We got the expected username in a single line but still the second part of that email id which is this domain name is still missing, we want to print this iitm.in as well as the second line of the output over here but because we are using this break statement it is stopping the execution of this for loop, we do not want to stop it, we want it to continue but we want to skip this particular letter. In such a case the keyword continue will help us.

(Refer Slide Time: 5:46)



```
# xyz123@iitm.in
email = input()
for c in email:
  if(c == '@'):
    print('')
    continue
  print(c, end = '')
```

```
xyz123@iitm.in
xyz123
iitm.in>
```

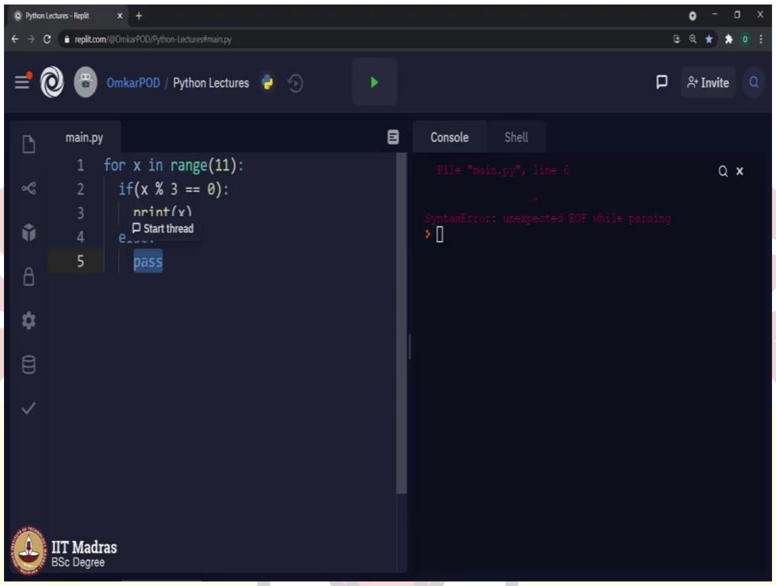
Let us try that continue, let us execute the code first and then we will discuss how exactly it works, the same input. Now, if you can observe we are getting the output xyz123, we skip the letter at the rate in between and then the remaining letters from email id, still it is not giving the expected output because as I said I want this particular domain name on the next line, that can be easily done by adding one print statement over here with empty quotes.

Let us execute again xyz123iitm.in, we got username as the first line and domain name as a second line. We got the required output but still we have to see how this particular keyword continue works. As soon as we reach to this character at the rate that is the place this if condition will become true then we are printing an empty string. As you remember we are writing this particular line only to shift that particular domain name to next line, so it does not have any specific relation to this particular keyword continue.

After this print computer will come to this particular keyword continue, this is the place where instead of stopping the entire for loop it will simply skip one iteration in between, which means whenever computer come across this particular keyword continue it simply jumps to next value in the given sequence.

In this case whenever computer reaches to at the rate it comes to continue and because of that it skips that iteration and moves to next character which is i, same thing for next i, then t, then m, then dot, again i and n. Because in all these cases this if condition will be false, hence, this block will not execute and we will print the required domain name on the next line.

(Refer Slide Time: 8:17)



```
for x in range(11):
  if(x % 3 == 0):
    print(x)
  else:
    pass
```

```
File "main.py", line 6

SyntaxError: unexpected EOF while parsing
›
```

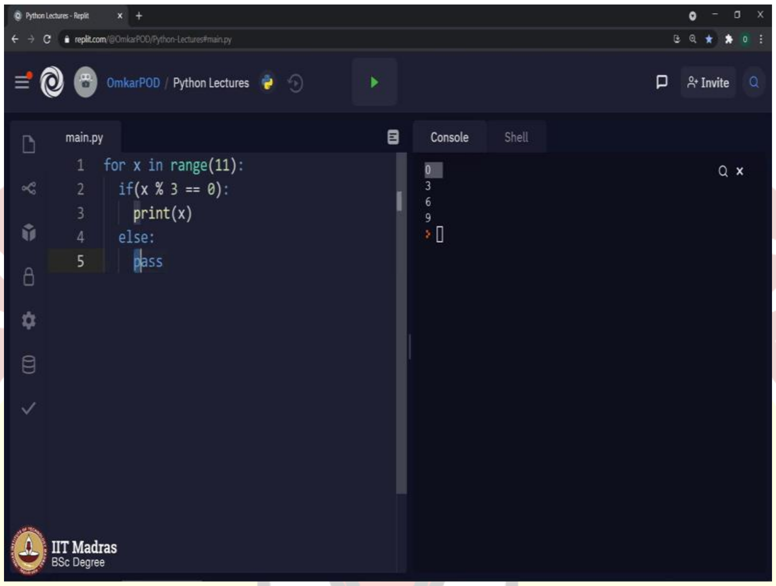
Let us move to next keyword which is pass. In order to demonstrate this particular keyword let me start with the problem statement, I want to categorize all the numbers from 1 to 10 into 2 categories. The first category; all the numbers which are divisible by 3, and second all the numbers which are not divisible by 3. And I want to print all those numbers which come under first category which is all the numbers which are divisible by 3.

And I am still not sure what to do with second category. I am sure you all can write this particular code. So, let us start with it. For x in range 11 because I want numbers from 0 to 10, if x mod 3 is equal equal to 0 print x, it will print all the numbers from 0 to 10 which are divisible by 3, perfect, so far it is correct.

But if you remember I mentioned I want to divide numbers into two categories which means if numbers are not divisible by 3, then I should do something with those numbers. But at this point of time I am not sure what I want to do with those numbers. Hence, I cannot write any specific code inside this else block. What will happen if I leave it blank and execute this code?

It says unexpected eof while parsing, do not think too much about this complicated terms eof and parsing, the point is you cannot leave else block blank as it is, which means we have to write something inside that block. But as I said at this point I am not sure what I want to do with those numbers, this is the place where we will use the keyword pass.

(Refer Slide Time: 10:30)



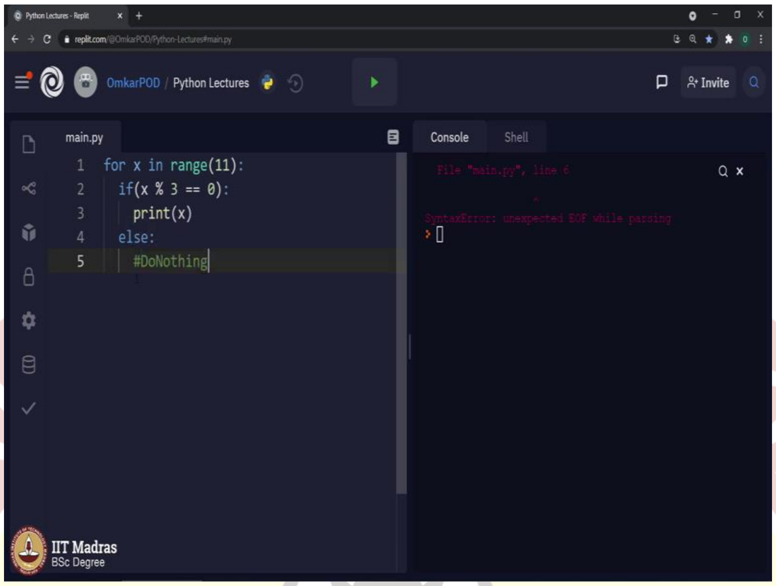
```
for x in range(11):
  if(x % 3 == 0):
    print(x)
  else:
    pass
```

```
0
3
6
9
>
```

Now, let us try to execute the code as you can see we are still getting the same output 0, 3, 6, 9 because of this print statement which is printing all the numbers which are divisible by 3. But when it comes to else it is not printing anything at all because this keyword pass is just a placeholder, it performs no operation, it is a null statement in Python. Because of that, when we started with 0, now 0 is divisible by 3, so print 0 we got the output.

Next when, the value of x became 1 and as 1 is not divisible by 3 we came across this else block and when computer reach this particular keyword pass it did nothing and went back to next value of x which is 2. Once again it came to pass, did nothing, went back to x, next value 3, we print the 3 and so on till 10. Now, you might wonder what is the difference between this keyword pass and a comment, because we have studied computer ignores the comment. In this case as well computer is behaving in the same manner.

(Refer Slide Time: 11:56)



```
for x in range(11):
  if(x % 3 == 0):
    print(x)
  else:
    #DoNothing
```

```
  File "main.py", line 6

            ^
SyntaxError: unexpected EOF while parsing
>
```

Let us see what happens if we write a comment, let us say do nothing, let us execute, still we are getting that same error unexpected eof while passing, which means there is a difference between comment and pass and the difference is whenever computer come across a comment, it simply ignores it and jumps to the next line.

But when computer come across this particular keyword pass, it will not ignore the keyword, it will execute the keyword but the meaning of this keyword is nothing because of that even though computer executes this particular keyword it will not perform any specific operation and it will jump to next line. Thank you for watching this lecture, happy learning.