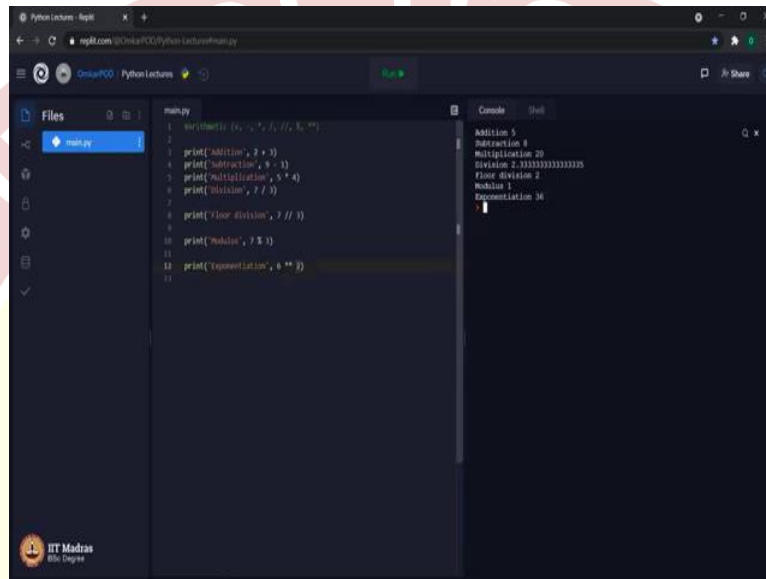




IIT Madras
ONLINE DEGREE

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
Operators and Expressions 2

(Refer Slide Time: 0:16)



```
main.py
1: result = (2+3)*4**3
2:
3: print('Addition', 2 + 3)
4: print('Subtraction', 9 - 1)
5: print('Multiplication', 5 * 4)
6: print('Division', 7 / 3)
7:
8: print('Floor Division', 7 // 3)
9:
10: print('Modulus', 7 % 3)
11:
12: print('Exponentiation', 4 ** 3)
13:
```

```
Console
Addition 5
Subtraction 8
Multiplication 20
Division 2.3333333333333335
Floor Division 2
Modulus 1
Exponentiation 64
```

Hello Python students. In last lecture we saw operators and expressions. We will continue from there and see some more different types of operators. In Python operators are divided into three major categories, first, arithmetic operators; second, relational operators and third, logical operators. Let us start with arithmetic operators. As you can see we have these many arithmetic operators in Python.

Addition, subtraction, multiplication, division, these four operators are familiar to all of us. First, let us see these four operators and then we will move to remaining three operators. Addition 2 plus 3, subtraction 9, so on, multiplication 5 multiplied 4, division 7 divide by 3. Let us execute the code as expected we are getting the output as 5, 8, 20, and 2.333 for the division. Now we will focus on remaining three operators.

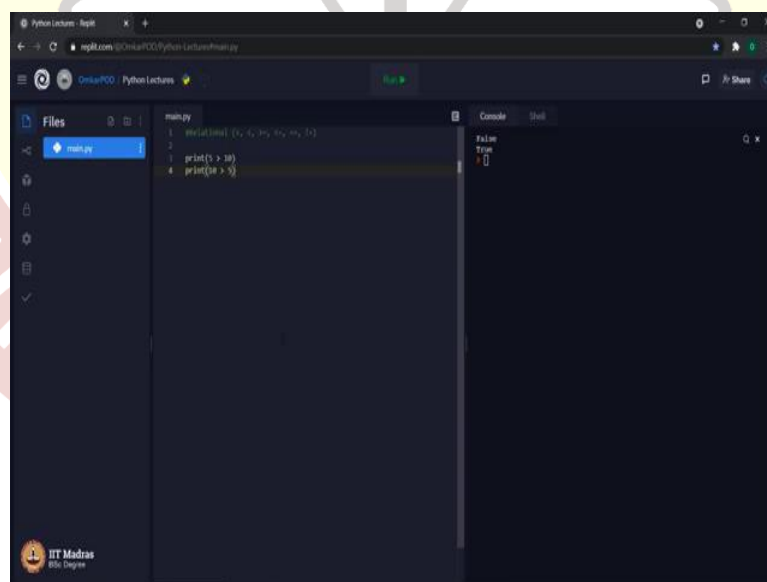
Out of that the first one is called floor division. This particular operator looks similar to division operator, but instead of one forward slash, we have two forward slash in floor division operator. Let us first execute the code, then we will see what is the difference between division and floor division.

As you can see over here the output for division operation was 2.333, which is a fractional value whereas for floor division it is an integer value and that is the difference between division operator and floor division operator. Whenever we divide 7 by 3, we get some fractional value, but when we divide 7 by 3 using a floor division operator it provides us only the integer part of that division, which is 2 in this case.

The next operator is called modulus operator. This operator is also related to division operation but it is different than division or floor division. Let us see how it works? This particular operator is represented using this percent symbol, 7 modulo 3. Let us execute first, output is 1. We are getting this output as 1 because whenever we divide 7 by 3, the remainder is 1, division operator gives us the output in fractional value.

Floor division divides the numbers and provides only the integer quotient part of it whereas modulus operator provides the remainder after the division. Now let us move to last operator which is exponential operator which is represented using this star star symbol. Exponential 6 raised to 2. As expected the output is 36, that is because 6 raised to 2 as in 6 power 2 is 36. All these different set of operators which you can see on your screen are called as arithmetic operators.

(Refer Slide Time: 4:07)



The screenshot shows a Python IDE with a file named 'main.py'. The code in the editor is as follows:

```
1 result = (10 + 2) * 3
2
3 print('1 + 10')
4 print('10 * 3')
```

The IDE also shows a 'Console' panel on the right, which is currently empty. The bottom left corner of the IDE displays the 'IIT Madras' logo and the text 'IIT Madras 600 075'.

```
Python Lectors - Replit
replit.com/@ChinaiPOD/Python-Lectors/main.py
Python Lectors
main.py
1 isNotEqual(x=1,y=2,z=3,w=4)
2
3 print(x < 10)
4 print(x < 5)
```

Console

True
False

HT Madras
BSc Degree

```
Python Lectors - Replit
replit.com/@ChinaiPOD/Python-Lectors/main.py
Python Lectors
main.py
1 isNotEqual(x=1,y=2,z=3,w=4)
2
3 print(x < 5)
4 print(x < 10)
```

Console

False
True

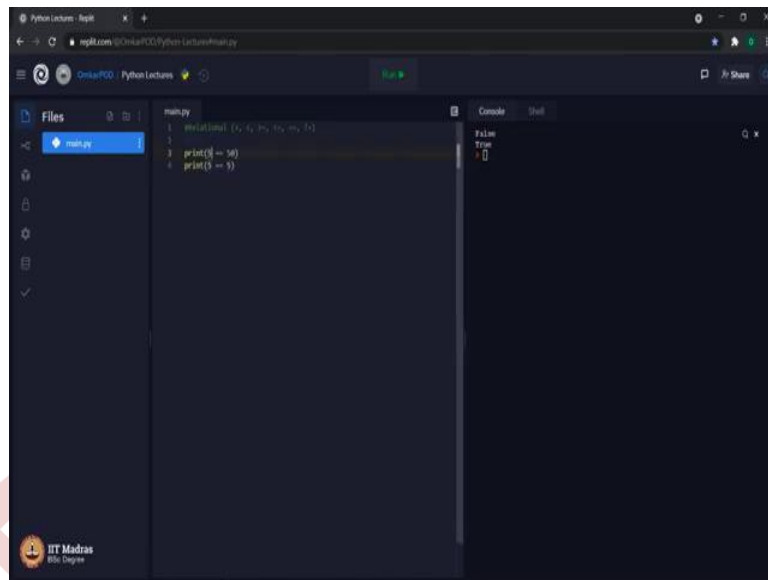
HT Madras
BSc Degree

```
Python Lectors - Replit
replit.com/@ChinaiPOD/Python-Lectors/main.py
Python Lectors
main.py
1 isNotEqual(x=1,y=2,z=3,w=4)
2
3 print(x < 5)
4 print(x < 10)
```

Console

False
True

HT Madras
BSc Degree

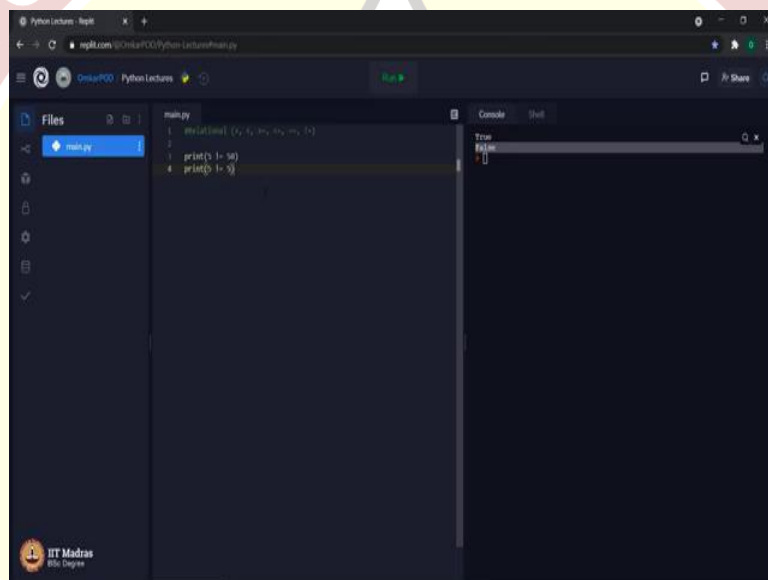


The screenshot shows a Python IDE with a file named 'main.py'. The code in the file is:

```
1 myList1 = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
2
3 print(8 >= 5)
4 print(5 >= 5)
```

The console output on the right shows:

```
True
True
```



The screenshot shows the same Python IDE with the same code as above. The console output on the right shows:

```
True
False
```

Now let us move to second set of operators. These operators are called relational operators, greater than, less than, greater than equal to, less than equal to, double equal to, and not equal to. If you remember you have studied all these operators in your previous course Computational Thinking. Let us go through these operators one by one to demonstrate how they are used in Python.

Let us say print 5 greater than 10, output is false, because 5 is not greater than 10, if we invert these two values and make it 10 greater than 5, then the output should be true, as you can see this greater than operator is only giving output in terms of Boolean value which can be either false or true. Let us execute the same code by changing the operator from greater than to less than, now 5 is less than 10.

So, this statement should be true whereas 10 is not less than 5, so this should be false. Let us see, true and false, which is what was expected. Now, the next operator is greater than equal to, now let us see what is the difference between greater than and greater than equal to. Let us make it 5 greater than 5 and 5 greater than equal to 5.

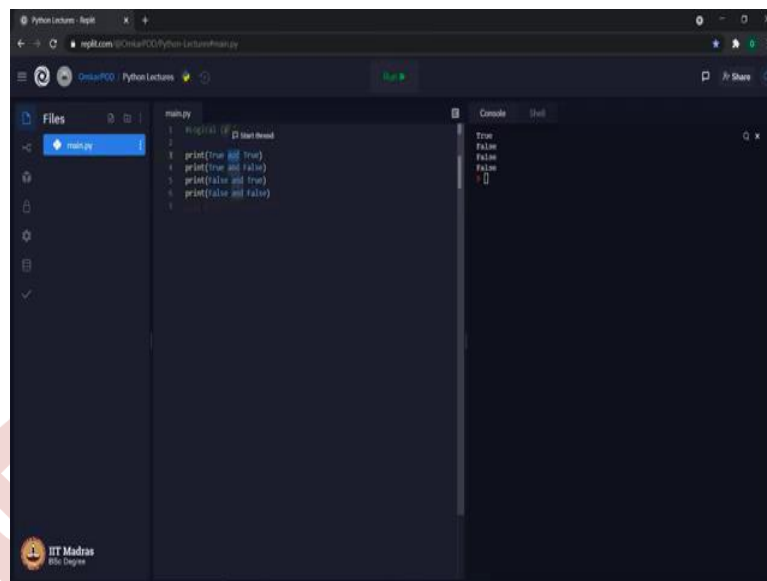
Let us execute and see the difference. In first case 5 is not greater than 5 because they are equal, that is why the output is false whereas in second case 5 is greater than equal to 5, in this case the greater than part may not be true but this equal part is true because of which output we are getting as true.

Similar is the case for less than equal to symbol. Let us try it less than less than equal to. Let us execute, same output, because once again 5 is not less than 5, but 5 is less than equal to 5. Once again this equal to part is true that is why we are getting output as false, followed by true. Next symbol is double equal to, let us use that, 5 double equal to 50, 5 double equal to 5, let us execute and see the difference. Now the output which we are getting is false followed by true. The first we are getting false because 5 is not equal to 50. The number 5 is not equal to the number 50.

At the same time the number 5 is equal to number 5 that is why we are getting true in second case and false in first case. This double equal to operator compares first operand with second operand and if they are equal, then it prints the value true and if they are not it will print the value false. The last operator is not equal to operator, which is exactly opposite of this double equal to operator. Over here if we use not equal to 50 and not equal to 5, then let us see what happens to the output.

Now we are getting true when we are saying 5 not equal to 50, which means this not equal to operator will compare operand one which is 5 with operand two which is 50, and if they are not equal then it will print true, whereas if they are equal then it will print false. As you must have noticed with respect to relational operators the output is always Boolean value, which means output of any expression using relational operator is always going to be a Boolean value.

(Refer Slide Time: 8:35)

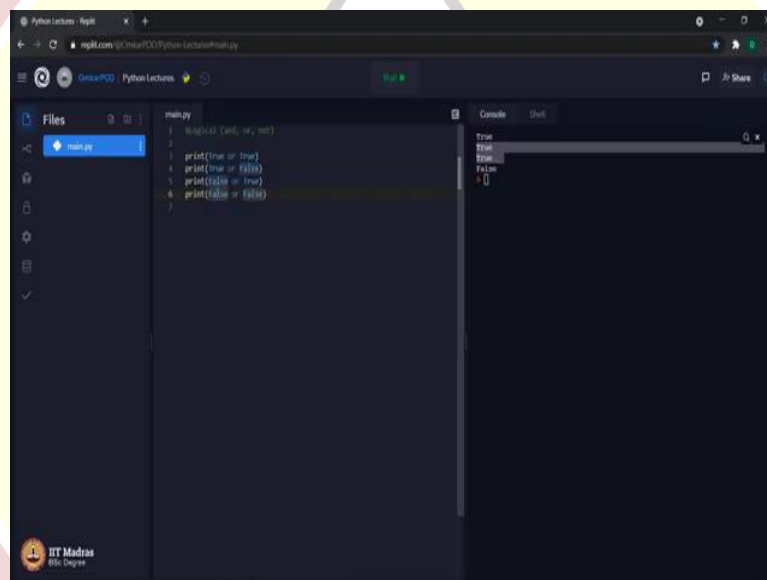


The screenshot shows a Python REPL interface with a file named `main.py` open. The code in the file is as follows:

```
1 logical (and, or, not)  
2  
3 print(true and true)  
4 print(true and false)  
5 print(false and true)  
6 print(false and false)  
7
```

The console output on the right shows the results of these operations:

```
True  
False  
False  
False
```

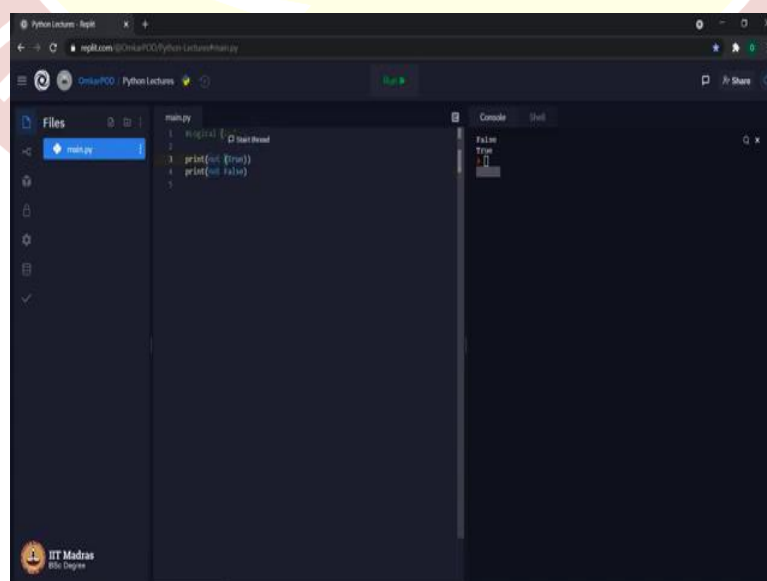


The screenshot shows a Python REPL interface with a file named `main.py` open. The code in the file is as follows:

```
1 logical (and, or, not)  
2  
3 print(true or true)  
4 print(true or false)  
5 print(false or true)  
6 print(false or false)  
7
```

The console output on the right shows the results of these operations:

```
True  
True  
False  
False
```



The screenshot shows a Python REPL interface with a file named `main.py` open. The code in the file is as follows:

```
1 logical (and, or, not)  
2  
3 print(not true)  
4 print(not false)  
5
```

The console output on the right shows the results of these operations:

```
False  
True
```

Next set of operators are called as logical operators. These are the logical operators and, or and not. Once again we have seen these operators before, but let us explore them using Python. First let us see the and operator print true and true, true and false, false and true, false and false. Over here we are using this and operator with all four possible combinations of true and false. Let us check the output first.

True, false, false, false, we are getting this output because true and true evaluates to be true, in all other cases the output is false, based on this we can conclude that when the and operator gets both its operand as true then and then only it gives the output as true. In all other cases the output is false. Now, let us replace the and operator with or and see the difference in the output.

Let us execute first true, true, true, and last is false. Now, in case of or operator expects at least one of its operand to be true, if that is the case the output is true otherwise the output will be false. Hence, in first case both operands are true, so output is true, the first operand is true, in third case the second operand is true, that is why in both these cases as well the output is true, whereas for last case both the operands are false as we cannot find even a single true on either side of our operator the output is false.

Now, let us see the last operator which is not operator. Let us execute this code, output is false followed by true, not of true is false, whereas not of false is true, which means this not operator simply inverts the Boolean value from true to false and from false to true. But you must have observed one difference in these two print statements.

In first print statement I have used not followed by a bracket inside which I have used a Boolean value true whereas in second print statement I have used not operators followed by false directly without using any parentheses, still we are getting the expected output which means the not operator can be used either with brackets or without brackets, still we get the expected output. Thank you for watching this lecture. Happy learning!