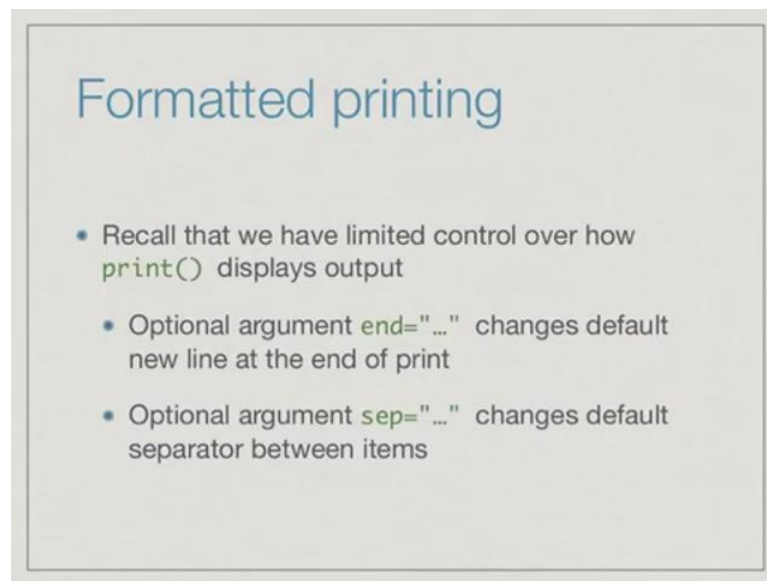**Programming, Data Structures and Algorithms in Python**
Prof. Madhavan Mukund
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Week - 05
Lecture - 05
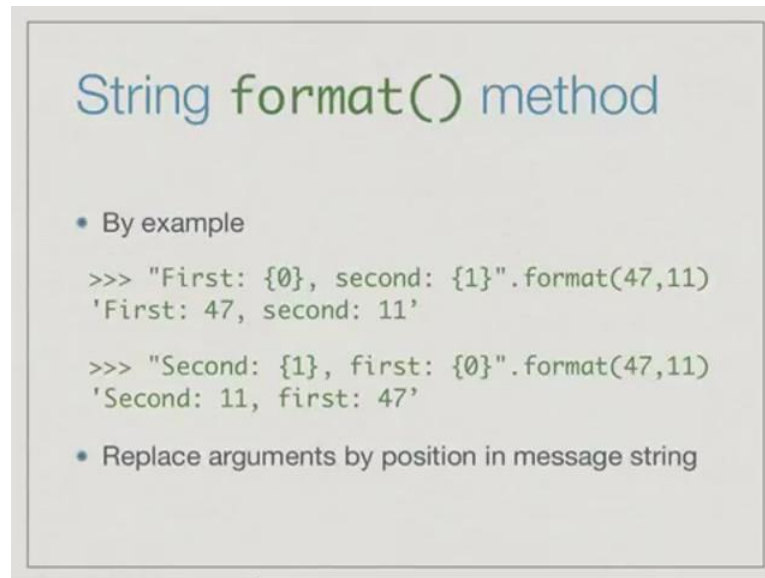**Formatting printed output**

(Refer Slide Time: 00:01)

When we looked at input and print earlier in this week's lectures, we said that we could control in a limited way how print displayed the output. Now by default print takes a list of things to print, separates them by spaces and puts a new line at the end.

However, it takes an optional argument end equal to string which changes what we put at the end, in particular if we put an empty string it means that it does not start a new line, so the next print will continue on the same line. Similarly we can change the separator from a space to whatever we want and in particular if we do not want any spaces we can put them ourselves and just say the separator is nothing - the empty string.

Now, sometimes you want to be a little bit more precise, so for this we can use the format method which is actually part of the string library. So the set of things that we can do with strings, last, in the previous lecture we looked at other things we can do string like, find, replace and all this things, so this is like that, it is in the same class.

Remember when you are doing print, you are actually printing a string. So, anything you can do to modify a string will give you another string that is what you are going to print. So, the string here is actually going to call a format method. So, the easiest way to do this is by example. We have a base string here, which is first, second and we have these two funny things in braces.

The funny things in braces are to be thought of as the equivalent of arguments in the function, these are things to be replaced by actual values and then what happens is that when you give this string and you apply the format method then the 0 refers to the first argument and 1 refers to the second argument. So what we are doing is, we are replacing by position, so if I actually take this string and I pass it to python the resulting thing is first colon 47, second colon 11, because the first argument, the brace 0 is replaced by the first argument to format 47 and the second replaces the second.

Now the positions determine the names so they do not have to be used in the same order. So, we could first print argument 1 and then print argument 0 as the second example

shows. Essentially, this version of format allows us to pass things into a string by their position in the format thing. So we are replacing arguments by position.

(Refer Slide Time: 02:34)

Now we can do the same thing by name. This is exactly like defining a function where remember, we said that we could give function arguments and we could pass it by name. So, in same way here we can specify names of the arguments to format, we can say f is equal to 47, s is equal to 11, that is first and second.

Now we can say one f and two s. Now here the advantage is not by position but by name. If I take these two things and I exchange them, so if I make f the second argument and s the first argument, and I pass it to the same string then f will be correctly caught as 47 and s as 11, so here by using the name not the position. So, the order in which you supply the things to format does not matter.

(Refer Slide Time: 03:20)

So, up to this point we have not done any formatting. All we have done is we have taken a string and we have told us how to replace values for place holders in the string. There is no real formatting which has happened because whatever we did with that we could have already done using the existing print statement that we saw.

Now the real formatting comes by giving additional instructions on how to display each of these place holders. So here we have 0 followed by some funny thing, we have this special colon and what comes after the colon is the formatting instruction. This has two parts here, we see a 3 and a d and they mean different things. So the 3d as a whole tells us how to display the value there is going to be passed here, that is the first thing. D is a code that specifies, I think it stands for decimal, so d specifies that 4 should be treated as an integer value. So, we should actually display it as a normal integer value namely it's a base ten integer.

Finally, 3 says that we must format 4 so that it takes three spaces. It occupies the equivalent of three spaces though it is a single digit. So if I do all this, then what happens is I get value, now notice that already there is one space here, then it going to take three spaces so I am going to get two blank spaces and then a 4, so that is why we have this long, so this is actually three blank spaces and then a 4. This whole thing, this part of it comes from the format. So I have a blank space, a blank space and a 4, because I was

told to put 4 in a width of 3 and think of it as a number, so since its number it goes to right hand.

(Refer Slide Time: 05:06)

Let us look at another example. Supposing, I had number which is not an integer, but a floating point number, so it is 47.523. Now here first thing is that we have instead of d we have f for floating point. So, f, 6.2f, this whole thing to the right of the colon tells me how to format the values comes from here. 6.2f breaks up as follows; the f, the letter part of it always tells me what the type of value is. So, it says that 47.523 should be treated as a floating point value. And the second thing is that it says this 6 tells me how much space I have to write whatever I have to write. So it says the total value including the decimal point, everything is going to be 6 characters wide.

Finally, the 2 says how many digits to show after the decimal point. If I apply all this then first of all because I have only two digits after decimal point this 3 gets knocked off, and then because it's set to use 6 character, now if I count from the right, this is 1, 2, 3, 4, 5 characters but it's set to use 6 characters, that is why there is an extra blank here. If you notice here, there is only one blank, but here there are two blanks. The second blank comes from the format statement.
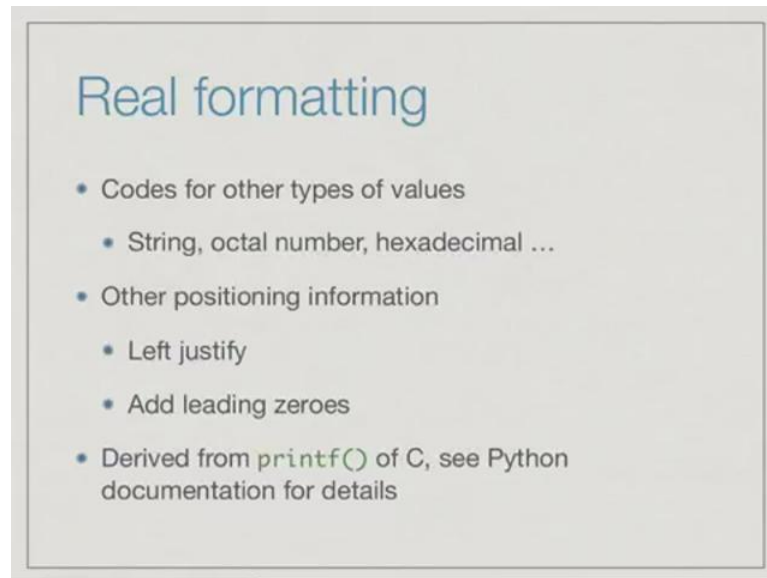
Unfortunately this is not exactly user friendly or something that you can easily remember, but there are codes for other values. We saw f and d, so there are codes like s for string, and o for octal, and x for hexadecimal. All these values can be displayed also using these formatted things. And you can also do other things you can tell it not to put it on the right put it on the left, so you can left justify the value. In a field of width 5 for example, if you want to put a string you might say the string should come from the left, not from the right. Then you can add leading zeroes, so you might to display a number 4 not in width 3 not as 4, but as 004. So, all these things you can do.

As I said this is a whole zoo of formatting things that you can do with this. These all have their origin from the language C and the statement called printf in C, so the exact format statements in our 0, 3d and 6.2f and all what they mean. It is best that you look up python documentation, you may not need all variations of it, the ones that you need you can look up when you need them, but it is useful to know that this kind of formatting can be done.