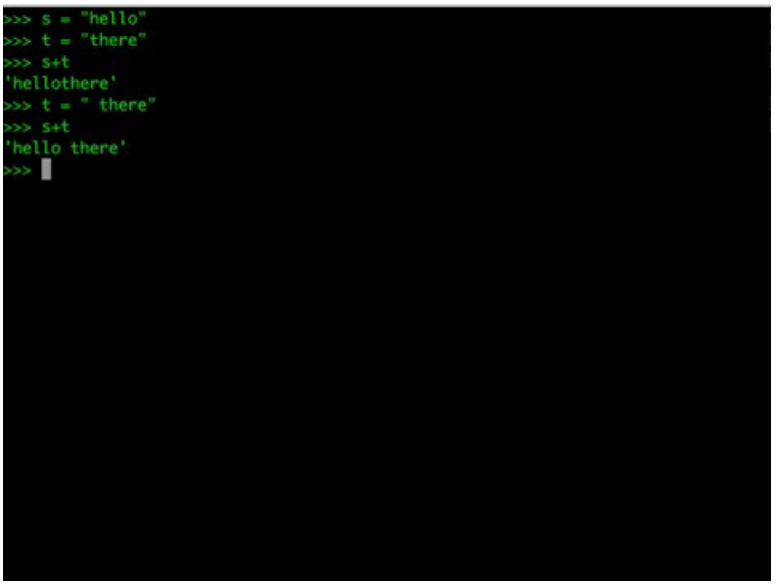
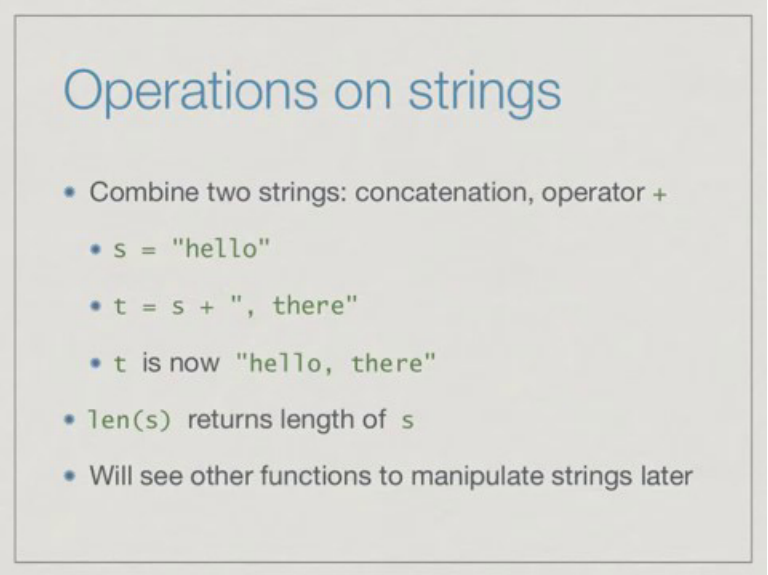
(Refer Slide Time: 11:52)




Let us look at an example in the interpreter. Just to emphasize one point; supposing I said s was hello and t was there, then s plus t would be the value hello there. Now notice that there is no space. So, plus literally puts s followed by t, it does not introduce punctuation, any separation, any space and this is as you would like it. If you want to put a comma or a space you must do that, so if you say t instead of that was space there t is the string consisting of blank space followed by there, now if I say s plus t, I get a space between hello and there.

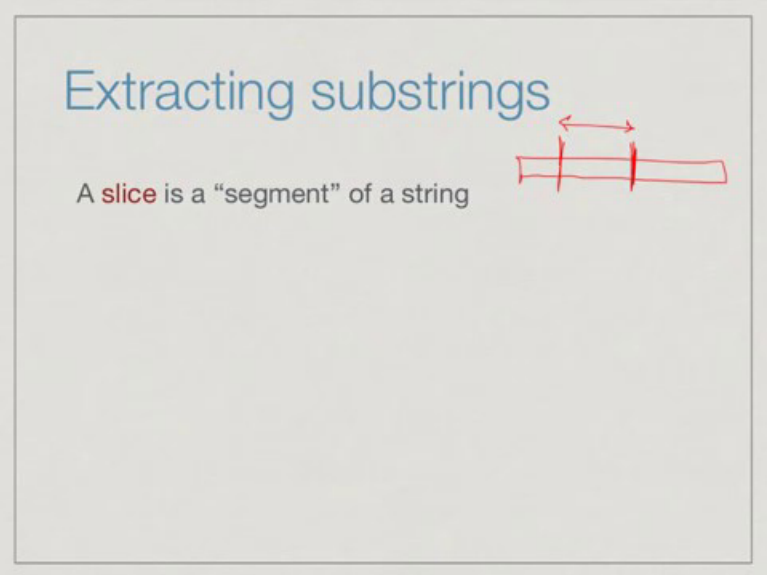
This is important to note that plus directly puts things together it does not add any punctuation or any separation between the two values. So, it is as though you have one new string which is composed of many old strings whose boundaries disappear completely.

(Refer Slide Time: 12:47)



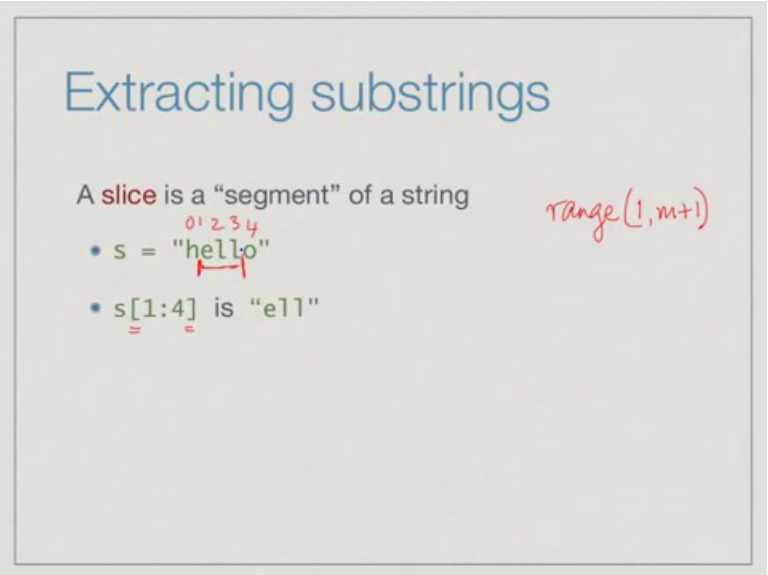

We can get length of the string using the function len. So, len(s) returns the length of s. So, this is the number of characters. So, remember that if the number of characters is n then the positions are 0 to n minus 1. So, the length of the string s here would be 5, the length of the string t here would be 5 plus 7 – 12. There are many other interesting functions that one can use to manipulate strings, you can search and replace things, you can find the first occurrence of something and so on, and we will see some of these later on, when we get into strings and text processing and reading data from files in more details.

(Refer Slide Time: 13:26)




A very common thing that we want to do with strings is to extract the part of a string. We might want to extract the beginning, the first word and things like that. The most simple way to do this in python is to take what is called a slice. Slice is a segment, a segment means I take a long string which I can think of as a list of character and I want the portion from some starting point to some ending point.

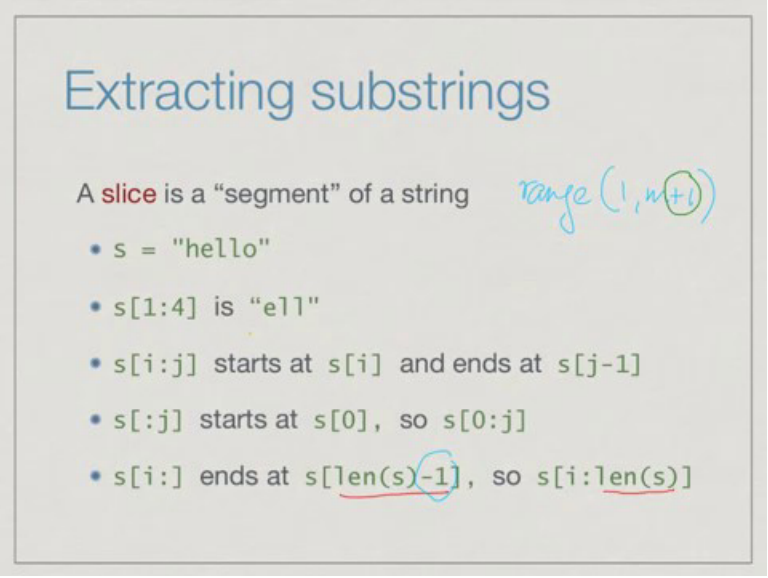
(Refer Slide Time: 13:55)



This is what python calls a slice. So, if we say s is hello as before, then for a slice we give this starting point and the ending point separated by colon. So, we use this square bracket notation exactly as though we were extracting part of a string, but the part that we are extracting is not the single position, but a range of positions from 1 to 4.

Now in python, we saw that we had this range function which we wrote last time, it said things like, if I want the numbers from 1 to m, I must write 1 to m plus 1, because the range function in python stops one position short of the last element of the range. So, in the same way, a slice stops one position short of the last index in the slice. So, if I do this then remember that hello has position 0, 1, 2, 3, 4, so the slice from 1 to 4 starts at 1 goes to 2, goes to 3, but does not go to 4, so it is only from e to l - the second l.

(Refer Slide Time: 14:59)



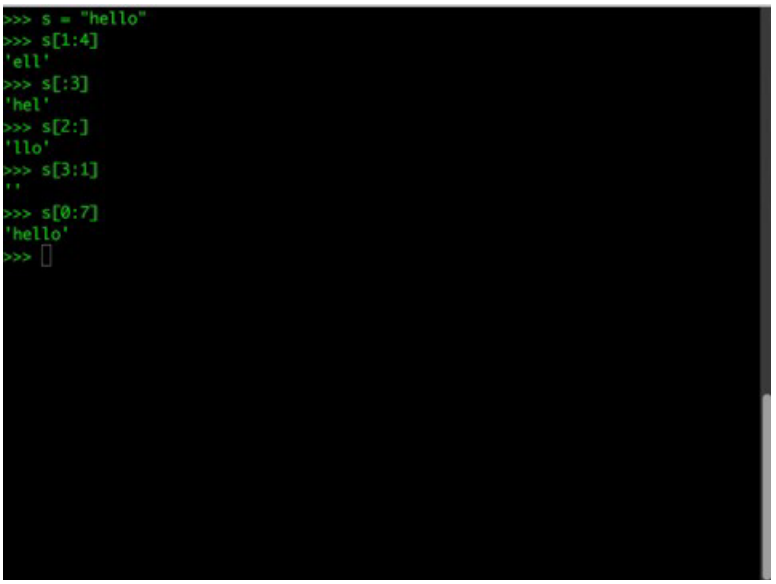

In general, if I write s i colon j then it starts at s i and ends at s j minus 1. There are some shortcuts which are easy to remember and use; very often you want to take the first n characters in the string, then you could omit the 0, and just say start implicitly from 0, so just leave it out, so just start say colon and j. So this will give us all position 0 1 up to j minus 1. So, if I leave out first position, it is implicitly starting from 0.

Similarly, if I leave out the last position it runs to the end of the string. So, if I want

everything from i onwards then I can say s i colon and this will go up to the position length of s minus 1, but if I write explicitly as a slice, I will only write length of s. So, essentially this is the main reason that python has this convention that whenever I write something like a range of 1 to m plus 1 then I have this extra plus 1 here. So, the main reason for this plus 1 here is to avoid having to write minus 1.

If I had to include the last character and if I start numbering at 0, then every time I wanted to go to the end of the string I would have to say length of s minus 1. It is much more convenient to just say length of s, and implicitly assume that it knows that it should not go to length of s, but length of s minus 1. So, this whole confusion if you would like to call that in python about that fact that all ranges end one short of the right hand side of the range, stems from the fact that you very often want to run from something to the length of it in a list or a sequence or a string and when you say that you do not want have keep remembering to say minus 1.

(Refer Slide Time: 16:45)

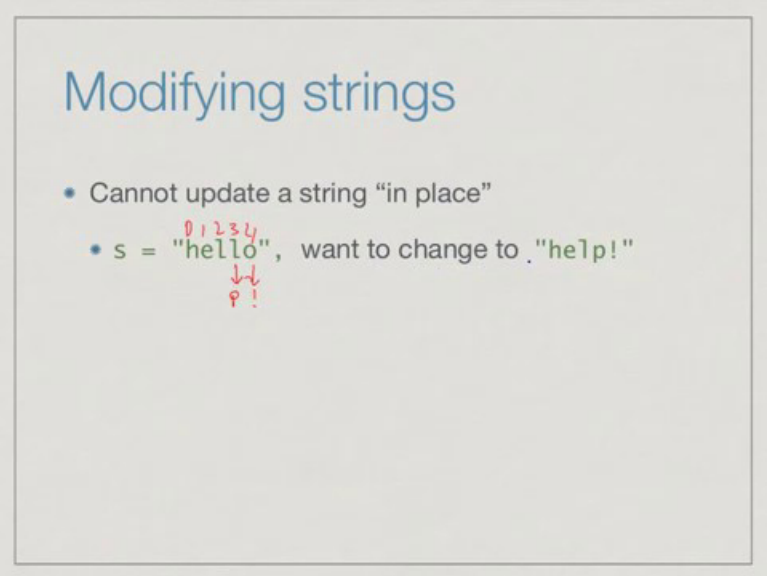



Let us play with the second in the python interpreter. So, if I say s is equal to hello then we saw that if I do 1 to 4, I get 'ell'. If I say colon 3 then I get 'hel' that is 0 1 2. If I say 2 colon, I get 'llo' that is 2 3 4. What if I say 3 2 1, so this says: start at position 3 and go up to position 1 minus 1 which is 0. So, python does not give you an error, it takes all these

invalid ranges, anything where for example, the starting point to the ending point does not define a valid range, and it says this is the empty string.

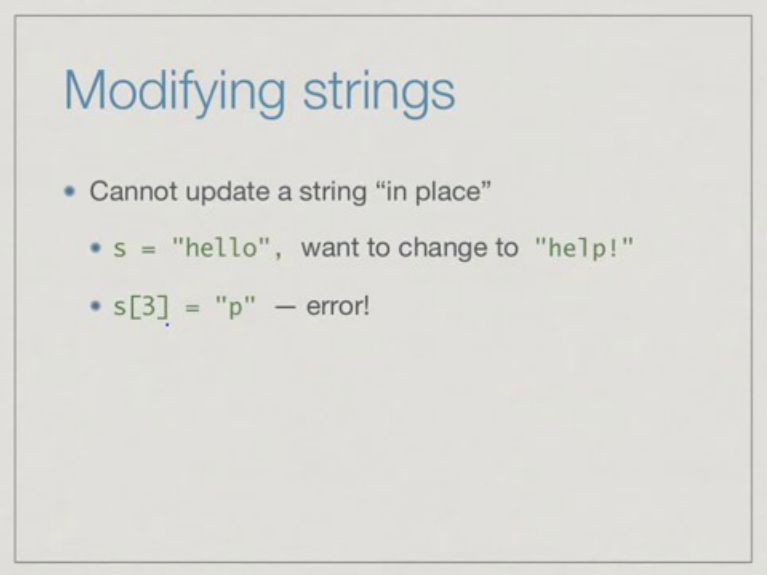
On the other hand, if I say something like go from 0 to 7, where there is no 7th position in the string, here python will not give an error instead, it will just go up to the last position which actually exists in the string below 7. So, in general these range values are treated in a sensible way, if you give values which do not make sense. As far as possible python tries to do something sensible with the slice definition.

(Refer Slide Time: 17:57)



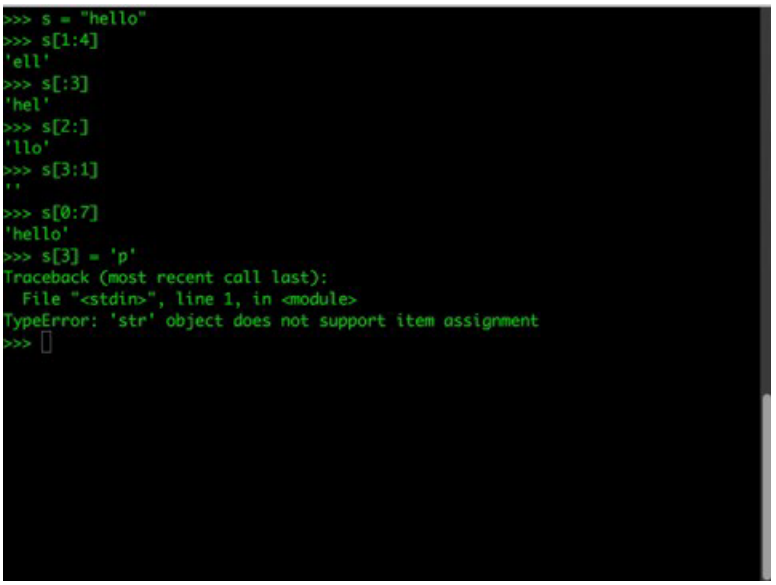

Though we have access to individual positions or individual slices or sequences within a string, we cannot take a part of a string and change it as it stands. So, we cannot update a string in place. Suppose, we want to take our string "hello" and change it to the string "help!" it would be nice if we could take the third and the fourth position. So, remember 0, 1, 2, 3, 4, 5, so 0, 1, 2, 3, 4, so it would be nice if we could say make this into a p and make this into an exclamation mark, so that I could get help instead of hello.

(Refer Slide Time: 18:36)



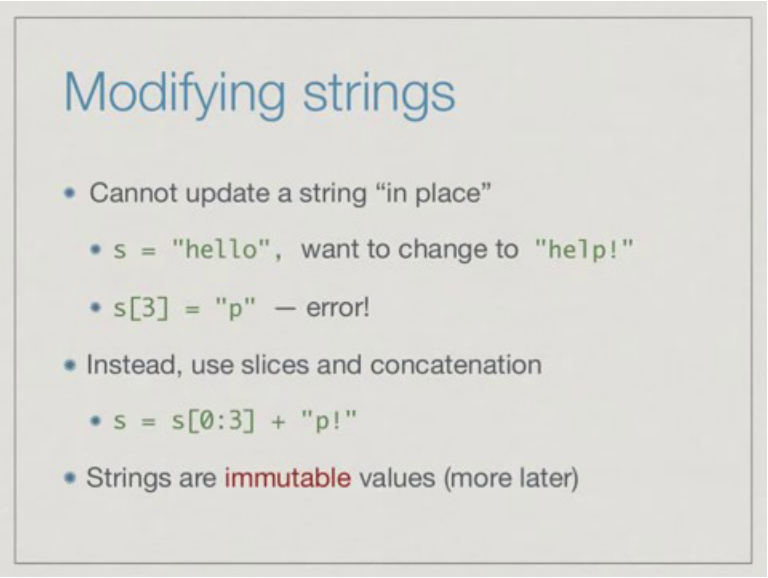

We would want to write something like change s 3, assign the value s 3 to be the string p. Now, unfortunately python does not allow this. So, you cannot update a string in place by changing its part. In fact, if you try this, you will actually get an error message, let us see.

(Refer Slide Time: 18:54)



Here we have the string hello defined in four, and if I now try to say s 3 is equal to p, then it says this does not support item assignment, which is what we are trying to say you cannot change parts of a string as it stands.

(Refer Slide Time: 19:12)



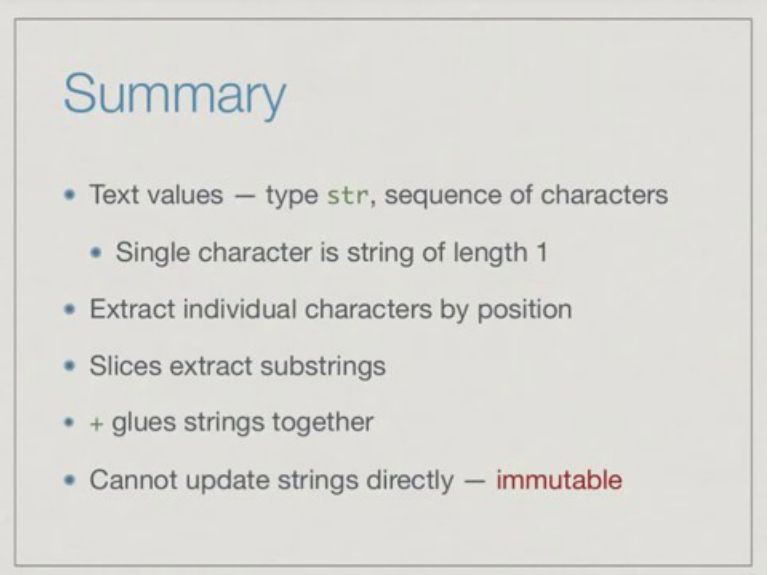

Instead of doing this, instead of trying to take a string and change the part of it as its stands what you need to do is actually construct a new string effectively using the notion of slices and concatenation. Here what we want to do is we want to take the first part of the string as it is. These are the first three characters, and then we want to change this to p exclamation mark. So, what we can say is update s by taking 0, 1, 2 which is slice 0 to 3 and concatenating it with the new string p exclamation mark. So, this is how you modify strings in python, but important thing is this is a new s we are not claiming that this s is same as old s.

There we build a new string from the old string and perhaps store it back in the same name, it is partly like when we say j is equal to j plus 5, we are actually saying that we have created a new value for j and stored it back in j.

Here again we are creating a new string and putting it back, but we are not modifying it. Now this distinction between modifying and creating a new value may not seem very

important at this moment, but it will become important as we go along. So, strings are what are called immutable values, you cannot change them without actually creating a fresh value; whereas, lists as we will see which are more general type of sequence can be changed in place you can take one part of a list and then replace it by something else. So, we will see more about this later, this is a fairly important concept. Remember for now that strings cannot be changed in place.

(Refer Slide Time: 20:42)




To summarize what we have seen is that text values are important for computation, and python has the types - string or str, which is a sequence of characters to denote text values. And there is no distinction for a separate type for a single character; there is no single character type in python, a single character is just a string of a length 1.

We can extract individual characters by index positions, we can use slices to extract sub strings, and we can glue strings together using the concatenation operator plus, but strings are immutable. We cannot take a value assigned to a string name and update it in place. We can create a new value by manipulating it using slices and concatenation, but we cannot directly update it, because strings are immutable.