Strings

CS303E: Elements of Computers and Programming

July 2, 2012

The String Data Type

- Represents text
- So far, we've used it for input and output
- Now, we'll take a closer look:
  - Representation
  - String operations
    - Like arithmetic operations, but different

String Representation

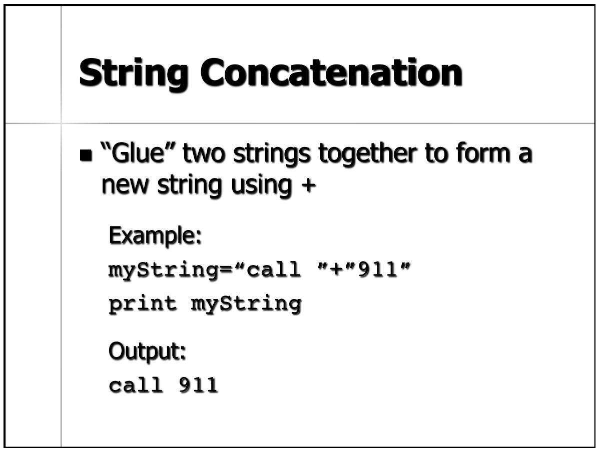
- Defined with single or double quotes
  - Need to start and end with same type of quote
    - `firstName='Steve'`
    - `lastName="Jobs"`
- Represented as a sequence of characters

S	t	e	v	e

Strings as Input

- Get using `raw_input()`
- OR `input()`, but then the user should include the text in quotation marks

- Best to use `raw_input()` and not expect the user to know to use quotation marks



String Concatenation

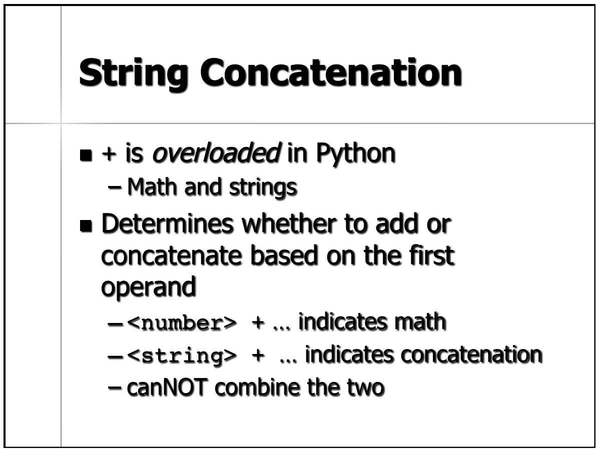
- "Glue" two strings together to form a new string using +

Example:

```
myString="call "+"911"
print myString
```

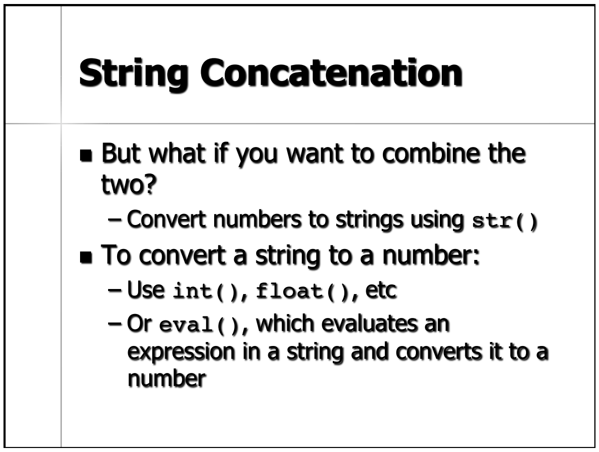
Output:

```
call 911
```



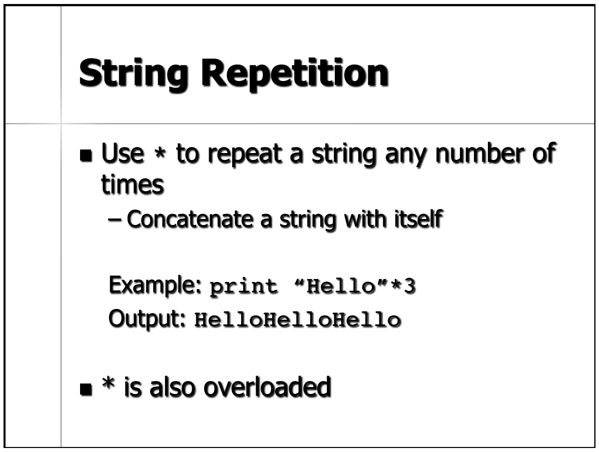
String Concatenation

- + is *overloaded* in Python
  - Math and strings
- Determines whether to add or concatenate based on the first operand
  - `<number> +` … indicates math
  - `<string> +` … indicates concatenation
  - canNOT combine the two



String Concatenation

- But what if you want to combine the two?
  - Convert numbers to strings using `str()`
- To convert a string to a number:
  - Use `int()`, `float()`, etc
  - Or `eval()`, which evaluates an expression in a string and converts it to a number



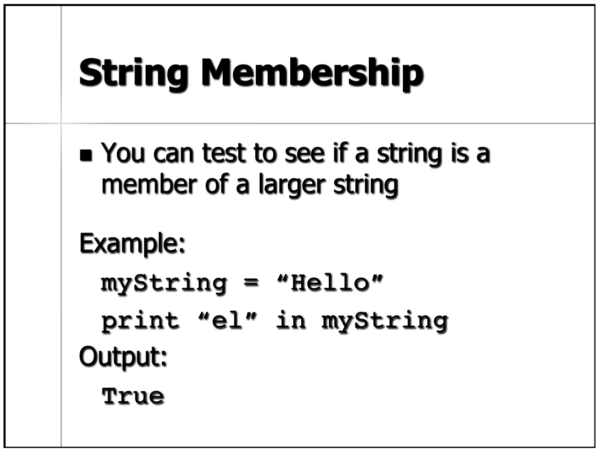
String Repetition

- Use `*` to repeat a string any number of times
  - Concatenate a string with itself

Example: `print "Hello"*3`

Output: `HelloHelloHello`

- * is also overloaded



String Membership

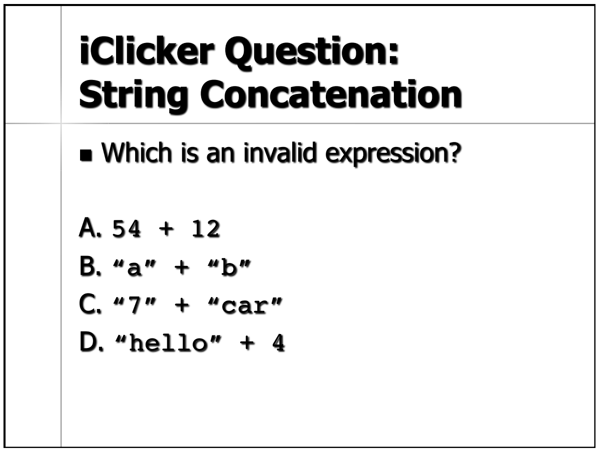
- You can test to see if a string is a member of a larger string

Example:

```
myString = “Hello”
print “el” in myString
```

Output:

```
True
```



iClicker Question: String Concatenation

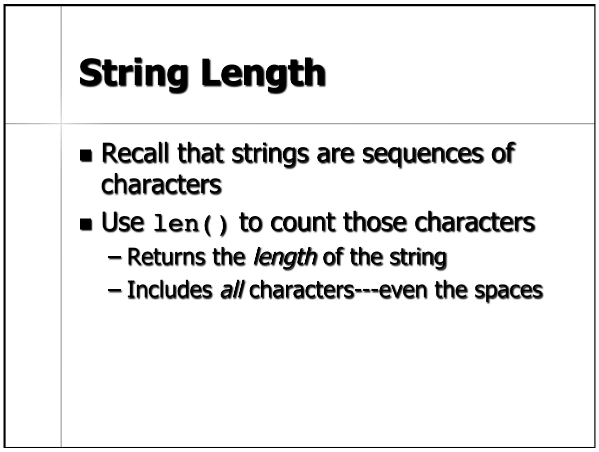
- Which is an invalid expression?

A. `54 + 12`

B. `“a” + “b”`

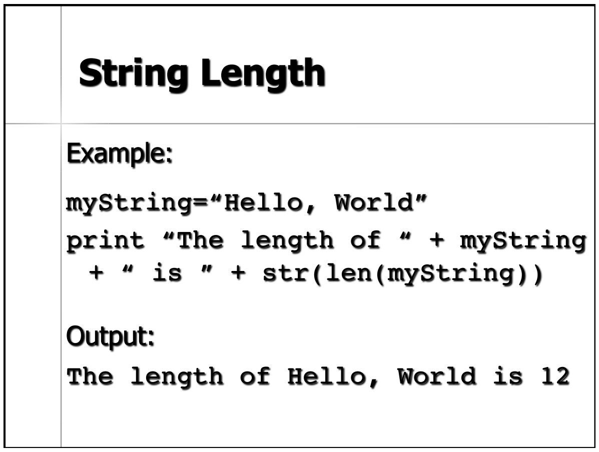
C. `“7” + “car”`

D. `“hello” + 4`



String Length

- Recall that strings are sequences of characters
- Use `len()` to count those characters
  - Returns the *length* of the string
  - Includes *all* characters---even the spaces



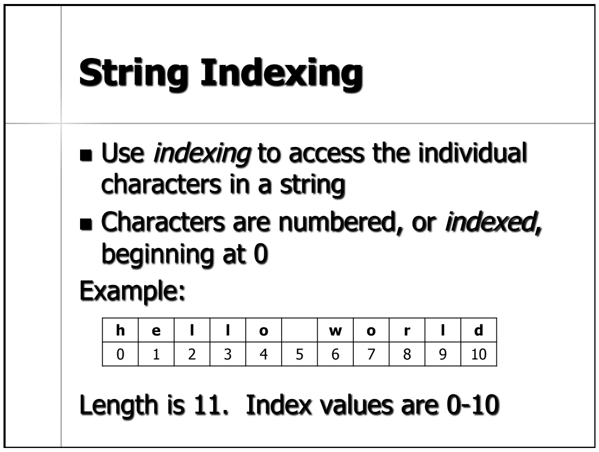
String Length

Example:

```
myString=“Hello, World”
print “The length of “ + myString
  + “ is ” + str(len(myString))
```

Output:

```
The length of Hello, World is 12
```



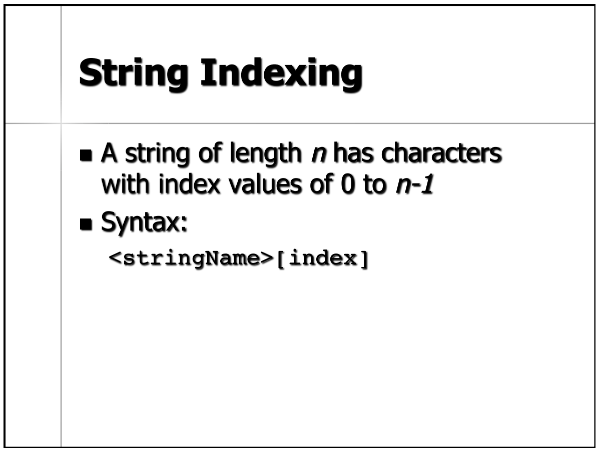
String Indexing

- Use *indexing* to access the individual characters in a string
- Characters are numbered, or *indexed,* beginning at 0

Example:

h	e	l	l	o		w	o	r	l	d
0	1	2	3	4	5	6	7	8	9	10

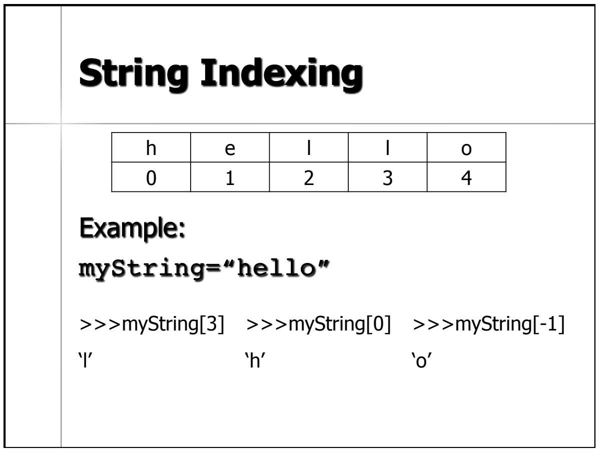
Length is 11. Index values are 0-10



String Indexing

- A string of length *n* has characters with index values of 0 to *n-1*
- Syntax:

```
<stringName>[index]
```



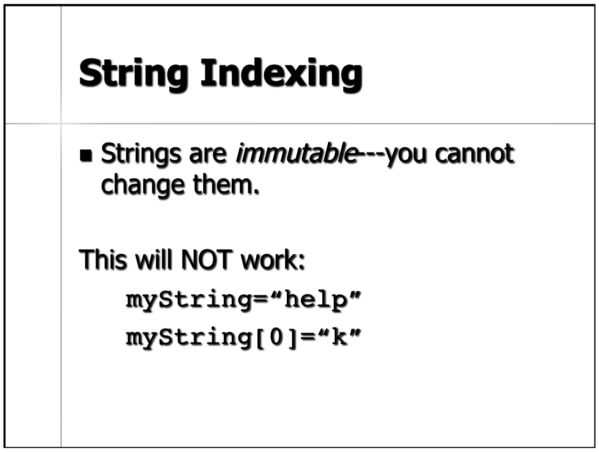
String Indexing

h	e	l	l	o
0	1	2	3	4

Example:

```
myString="hello"
```

```
>>>myString[3]   >>>myString[0]   >>>myString[-1]
'l'              'h'              'o'
```



String Indexing

- Strings are *immutable*---you cannot change them.

This will NOT work:

```
myString="help"
myString[0]="k"
```

iClicker Question: String Indexing

- In the string, “Monday”, what is the index of ‘d’?

A. 3

B. 4

C. 5

Strings and for Loops

- Once we can index into a string, we can use a for loop to iterate through each character:

```
myString=“hello”
for i in range(len(myString))
    print myString[i]
```

Strings and for Loops: Exercise

- Print the characters in a string in reverse

Iterating Over a String: Take Two

- Again, strings are sequences of characters

h	e	l	l	o

- So we can iterate…

Example:

```
myString=“hello”
for ch in myString
  print ch
```

Output:

```
h
e
l
l
o
```