Strings

Objectives

- Gain understanding of strings
- Learn multiple ways of creating strings
- Get string input from the user
- Store strings into variables
- Learn string concatenation
- Get characters of a string
- Convert between strings and integers

Remember?

Remember our hello world program:

```
print "hello world"
```

`"hello world"` is a **string**. We use " to tell Python to enclose the string. Note that it is not part of the string which is why it's not printed.

Exercise. What do you think is wrong with this statement:

```
print "She said, "Knowledge is power.""
```

If you don't see it, run it in Python.

Which of the following statements work?

```
print "this is a string"
print 'so is this'
print """and this"""
print '''one last time!!!'''
```

So now you know **four** different ways to enclose a string. Why do we need so many? Try this:

```
print 'She said, "Knowledge is power."'
```

Compare this with the above:

```
print "She said, "Knowledge is power.""
```

Make sure you see the difference!

String Variables and raw_input

Remember variables?

Exercise. Create a variable `age` and set it to your age. Next print the value of `age`.

Actually you can put strings into variables too. Try this:

```
name = "John Doe"
print name
```

Remember you can also get an integer value from the keyboard.

Exercise. Write a program that prompts the user for his age and print it out. Don't remember how to get an integer value and give it to a variable? No problem. Check your notes.

It's not too surprising that you can also get a string from the keyboard. Here's how you do it:

```
name = raw_input("Enter your name: ")
print "hi", name, "!!!"
```

Operators for Strings

You know that you can add integers. There are other operators.

Exercise. What are the operators for integer values? Do you recall the precedence rules? If not check the previous set of notes.

It turns out that you can use + on strings too. See if you can guess what + does on strings with these examples:

```
     print "abc" + "def"
```

At this point it's **important** to see the **difference** between the **integer** `1` and the **string** `"1"`. Try these statements:

```
     print 1 + 1
     print "1" + "1"
```

Make sure you understand why you get these outputs!

IMPORTANT JARGON. The + used for strings is called the **concatenation** operator.

Exercise. Does the following work?

```
     print "I am " + 24 + " years old"
```

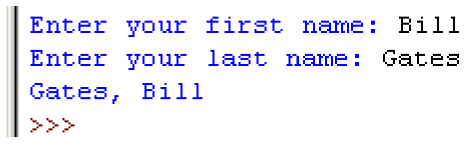
Answer: ______

The above tells you that you cannot add strings and integers.

Of course just like you can apply + to more than one value (example: 1 + 2 + 3 + 4 + 5), you can also apply + to several strings:

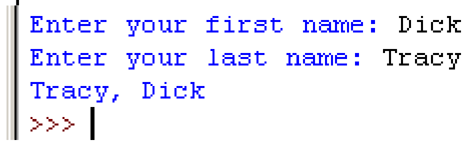
```
     print "John" + " " + "Doe"
```

Exercise. Write a program that when executed with the following inputs behaves as shown:



```
Enter your first name: Bill
Enter your last name: Gates
Gates, Bill
>>>
```

Here's another execution of the same program but with different inputs:



```
Enter your first name: Dick
Enter your last name: Tracy
Tracy, Dick
>>>
```

[Hint: Create a new variable and set it to an appropriate value. You need to use string concatenation twice.]

String Operators

The string is somewhat different from the integer because the string is made up of smaller units. For instance the string

```
name = "John Doe"
```

is made up of `J` and `o` and `h` and `n`, etc. Each of these smaller "things" are called characters. You can in fact pull out each of these characters in the following way:

```
name = "John Doe"
print name[0]
print name[1]
print name[2]
print name[3]
print name[4]
print name[5]
print name[6]
print name[7]
```

Exercise. Continuing the above, what happens when you execute

```
print name[8]
```

Of course you can do this:

```
name = "John Doe"
a = name[0]
print a
```

Converting Between Integers and Strings

Sometimes you want to convert the integer `42` to the string `"42"`.

Now why would you want to something like that? Here's one reason. Suppose you want to write a game. Obviously you need to keep integer values around such as scores. You do want to add. So the values *are* integers. However when you display the score onto the playing window, usually you can only display a string and not an integer. This is the case for the game library we are using. That's why.

OK. So how do you change a string made up of digits into an integer, such as changing “42” to 42. Try this example:

```
x = 1
y = str(x)
a = x + x
b = y + y
print "a: ", a
print "b: ", b
```

`What about the other way round? What if I have a string and I want to change that into an integer? Try this:`

```
x = "1"
y = int(x)
a = x + x
b = y + y
print "a: ", a
print "b: ", b
```