Computer Science 1 — CSci 1100
Lecture 4 — Python Strings

Reading

This material is drawn from Chapters 3 of *Practical Programming* and Chapter 8 of *Think Python*.

More Than Just Numbers

- Much of what we do today with computers revolves around text:
  - Web pages
  - Facebook
  - Text message

  These require working with *strings*.
- Strings are our third type, after integers and floats.
- We've already started to use strings in our output, for example,

```
def area_and_volume(radius, height):
    print "For a cylinder with radius", radius, "and height", height
    print "The surface area is ", area_cylinder(radius,height)
    print "The volume is ", volume_cylinder(radius, height)
```

Topics for Today

- String basics
- String operations
- Input to and (formatted) output from Python programs

Strings — Definition

- A string is a sequence of 0 or more characters delimited by single quotes or double quotes.

```
'Rensselaer'
"Albany, NY"
'4 8 15 16 23 42'
''
```

- We can print strings:

```
>>> print "Hello, world!"
Hello, world!
```

- Strings may be assigned to variables:

```
>>> s = 'Hello'
>>> t = "Good-bye"
>>> print s
Hello
>>> t
'Good-bye'
```

- Notice that unlike integers and floats there is now a difference between asking Python for the value of the variable and printing the variable!

Combining Single and Double Quotes in a String

- A string that starts with double quotes must end with double quotes, and therefore we can have single quotes inside.
- A string that starts with single quotes must end with single quotes and therefore we can have double quotes inside.
- To illustrate this, we will take a look at

```
>>> s = 'He said, "Hello, World!"'
>>> t = "Many single quotes here '''''''  and here '''  but still correct."
```

Multi-Line Strings

- Ordinarily, strings do not extend across multiple lines, causing an error if you try.
- But, starting and ending a string `"""` or `'''` tells Python to allow the string to cross multiple lines.
  - Any character other than `'''` (or `"""`, if that is how the string started) is allowed inside the string.
- Example,

```
>>> s1 = """This
is a multi-line
string."""
>>> s1
'This\nis a multi-line\nstring.'
>>> print s1
This
is a multi-line
string.
>>>
```

- Notice the `\n` when we ask Python for the value of the string (instead of printing it). This is an *escape character*, as we will discuss next.

Escape Characters

- Inserting a `\` in the middle of a string tells Python that the next character will have special meaning (if it is possible for it to have special meaning).
- Most importantly:
  - `\n` — end the current line of text and start a new one
  - `\t` — skip to the next "tab stop" in the text. This allows output in columns
  - `\'` — do not interpret the `'` as a string delimiter
  - `\"` — do not interpret the `"` as a string delimiter
  - `\\` — put a true back-slash character into the string
- We'll explore the following strings in class

```
f
>>> s0 = "*\t*\n**\t**\n***\t***\n"
>>> s1 = "I said, \"This is a valid string.\""
```

Exercise Set 1

1. Which of the following are valid strings? Fix the mistakes to make them all valid.

```
>>> s0 = "Sheldon Cooper's apartment is in Pasedena"

>>> s1 = 'This cheese shop's cheese is all gone"

>>> s2 = """We are
"The Knights of the Round Table"
"""

>>> s3 = "Toto, I said,\n"We aren't in Kansas, anymore!"


>>> s4 = 'Have you seen the "Incredibly Photogenic Guy"'s picture?'


>>> s5 = "Have you seen the 'Incredibly Photogenic Guy''s picture?"
```

2. What is the output?

```
>>> s = "Cats\tare\n\tgood\tsources\n\t\tof\tinternet\tmemes"
>>> print s
```

String Operations — Concatenation

- Concatenation: Two (or more) strings may be concatenated to form a new string, either with or without the + operator. We'll look at

```
>>> s0 = "Hello"
>>> s1 = "World"
>>> s0 + s1
>>> s0 + ' ' + s1
>>> 'Good' 'Morning' 'America!'
>>> 'Good ' 'Morning ' 'America!'
```

- Notice that

```
>>> s0 = "Hello"
>>> s1 = " World"
>>> s0 s1
```

  is a syntax error but

```
>>> "Hello" " World"
```

  is not. Can you think why?

String Operations — Replication

- You can replicate strings by multiplying them by an integer:

```
>>> s = 'Ha'
>>> print s * 10
HaHaHaHaHaHaHaHaHaHa
```

- What do you think multiplying a string by a negative integer or 0 does? Try it.
- Many expressions you might try to write involving strings and either ints or floats are illegal Python, including the following:

```
>>> 8 * 'Hello'
>>> 'Hello' * 8.1
>>> '123' + 4
```

  Think about why

String Operations — Functions

- You can compute the length of a string with `len`.
- You can convert an integer or float to a string with `str`.
- You can convert a string that is in the form of an integer to an integer using `int`
- You can convert a string that is in the form of a float to a float using, not surprisingly, `float`
- We will look at examples of all of these during lecture.

Exercise Set 2: String Operations

1. What is the output of the following:

```
>>> len('George')
>>> len(' Tom  ')
>>> s = """Hi
mom!
"""
>>> len(s)
```

2. Which of the following are legal? For those that are, show what Python outputs.

```
>>> 'abc' + str(5)
>>> 'abc' * str(5)
>>> 'abc' + 5
>>> 'abc' * 5
>>> 'abc' + 5.0
>>> 'abc' + float(5.0)
>>> str(3.0) * 3
```

3. Write a line of code that prints 50 `'*'` characters.
4. Write a function that takes a string as an argument and prints the string underlined with `=` equal to the length of the string. For example, we should have the following output:

```
underline('Tom')
print
underline('Super Bowl')
```

  should output

```
Tom
===

Super Bowl
==========
```

Use the `len` function and string replication.

String Output

- We already know a bit about how to use `print`, but here are a few things to remember.
  - A space is added between each value that is output in a `print` statement
  - Each `print` statement starts a new line of output... unless the previous `print` statement ended with a `,`
- But, let's look at some nicer ways to create output...

Formatted Output

- In the Lecture 3 Python program `area_volume.py`, the last few lines are

```
def area_and_volume(radius, height):
    print "For a cylinder with radius", radius, "and height", height
    print "The surface area is ", area_cylinder(radius,height)
    print "The volume is ", volume_cylinder(radius, height)

area_and_volume(5,10)
```

- This produces the output

```
For a cylinder with radius 5 and height 10
The surface area is  471.2385
The volume is  785.3975
```

- Here is better formatting, without the insignificant values

```
def area_and_volume(radius, height):
    print "For a cylinder with radius %d height %d" %(radius,height)
    print "The surface area is %.2f" %area_cylinder(radius,height)
    print "The volume is %.2.f" %volume_cylinder(radius, height)

area_and_volume(5,10)
```

which produces

```
For a cylinder with radius 5 height 10
The surface area is 471.24
The volume is 785.40
```

- We will discuss the significance of
  - `%d`
  - `%.2f`
  - `%(radius,height)`

User Input

- Python programs can ask the user for input using the function call `raw_input`.
- This waits for the user to type a line of input, which Python reads as a string.
- This string can be converted to an integer or a float (as long as it is properly an int/float).
- Here is a toy example

```
print "Enter a number"
x = float(raw_input())
print "The square of %.1f is %.1f" %(x,x*x)
```

- We can also insert the string right into the `raw_input` function call:

```
x = float(raw_input("Enter a rwo number"))
print "The square of %.1f is %.1f" %(x,x*x)
```

- We will use this idea to modify our area and volume calculation so that the user of the program types in the numbers.
  - The result is more useful and feels more like a real program (albeit one run from the command line).
  - It will be posted on the course website.

Summary

- Strings represent character sequences — our third Python type
- String operations include addition (concatenate) and replication
- We can concatenate by '+' or by using formatted strings:

```
>>> 'a' + 'b'
>>> '%d eggs and %s spam' %(2,'no')
```

- Functions on strings may be used to determine length and to convert back and forth to integers and floats.
- Escape sequences change the meaning of special Python characters or make certain characters have special meaning.
- Some special characters of note: `\n` for new line, `\t` for tab. They all precede with `\`
- We can read input using `raw_input()`