Lecture 3

Strings & Modules
Labs this Week

• Lab 1 is due at the beginning of your lab
  ▪ If it is not yet by then, you cannot get credit
  ▪ Only exception is for students who added late
    (Those students should talk to me)

• Should spend time entirely on Lab 2
  ▪ Similar format to last week
  ▪ Next weeks lab will be a bit longer
Readings for Next Few Lectures

Thursday Reading

• Sections 3.1-3.4
• Sections 8.1, 8.2, 8.4, 8.5
• Browse the Python API
  ▪ Do not need to read all of it
  ▪ Look over built-in functions

Next Week

• Complete Chapter 3

9/1/15 Strings & Functions
String: Text as a Value

- String are quoted characters
  - 'abc d' (Python prefers)
  - "abc d" (most languages)

- How to write quotes in quotes?
  - Delineate with “other quote”
  - **Example**: " ' " or ' " '
  - What if need both " and '?

- **Solution**: escape characters
  - Format: \ + letter
  - Special or invisible chars

<table>
<thead>
<tr>
<th>Char</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;</td>
<td>double quote</td>
</tr>
<tr>
<td>'</td>
<td>single quote</td>
</tr>
<tr>
<td>\n</td>
<td>new line</td>
</tr>
<tr>
<td>\t</td>
<td>tab</td>
</tr>
<tr>
<td>\</td>
<td>backslash</td>
</tr>
</tbody>
</table>
String are Indexed

- $s = 'abc d'$
  
  0 1 2 3 4  
  a b c d

- Access characters with []
  - $s[0]$ is 'a'
  - $s[4]$ is 'd'
  - $s[5]$ causes an error
  - $s[0:2]$ is 'ab' (excludes c)
  - $s[2:]$ is 'c d'

- Called “string slicing”

- $s = 'Hello all'$
  
  0 1 2 3 4 5 6 7 8  
  H e l l o  a l l

- What is $s[3:6]$?
  
  A: 'lo a'
  B: 'lo'
  C: 'lo '
  D: 'o '
  E: I do not know
String are Indexed

- \( s = 'abc d' \)

<table>
<thead>
<tr>
<th></th>
<th>( \text{a} )</th>
<th>( \text{b} )</th>
<th>( \text{c} )</th>
<th>( \text{d} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>( )</td>
</tr>
<tr>
<td>1</td>
<td>( )</td>
<td></td>
<td></td>
<td>( )</td>
</tr>
<tr>
<td>2</td>
<td>( )</td>
<td>( )</td>
<td></td>
<td>( )</td>
</tr>
<tr>
<td>3</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
</tr>
<tr>
<td>4</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
</tr>
</tbody>
</table>

- Access characters with \([\]\)
  - \( s[0] \) is 'a'
  - \( s[4] \) is 'd'
  - \( s[5] \) causes an error
  - \( s[0:2] \) is 'ab' (excludes c)
  - \( s[2:] \) is 'c d'

- Called “string slicing”

- \( s = 'Hello all' \)

<table>
<thead>
<tr>
<th></th>
<th>( \text{H} )</th>
<th>( \text{e} )</th>
<th>( \text{l} )</th>
<th>( \text{l} )</th>
<th>( \text{o} )</th>
<th>( \text{a} )</th>
<th>( \text{l} )</th>
<th>( \text{l} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
</tr>
<tr>
<td>1</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
</tr>
<tr>
<td>2</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
</tr>
<tr>
<td>3</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
</tr>
<tr>
<td>4</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
</tr>
<tr>
<td>5</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
</tr>
<tr>
<td>6</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
</tr>
<tr>
<td>7</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
</tr>
<tr>
<td>8</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
</tr>
</tbody>
</table>

- What is \( s[3:6] \)?
  
  A: 'lo a'
  B: 'lo'
  C: 'lo'  CORRECT
  D: 'o '  
  E: I do not know
**String are Indexed**

- \( s = 'abc\ d' \)
  
  \[
  \begin{array}{cccc}
  0 & 1 & 2 & 3 & 4 \\
  a & b & c & d \\
  \end{array}
  \]

- **Access characters with []**
  - \( s[0] \) is 'a'
  - \( s[4] \) is 'd'
  - \( s[5] \) causes an error
  - \( s[0:2] \) is 'ab' (excludes c)
  - \( s[2:] \) is 'c d'

- Called “string slicing”

- \( s = 'Hello all' \)
  
  \[
  \begin{array}{cccccccc}
  0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
  H & e & l & l & o & \text{all} \\
  \end{array}
  \]

- **What is \( s[:4] \)?**
  
  A: 'o all'
  B: 'Hello'
  C: 'Hell'
  D: Error!
  E: I do not know

9/1/15

Strings & Functions 7
String are Indexed

- \( s = 'abc d' \)

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td></td>
</tr>
</tbody>
</table>

- Access characters with \([\ ]\)
  - \( s[0] \) is 'a'
  - \( s[4] \) is 'd'
  - \( s[5] \) causes an error
  - \( s[0:2] \) is 'ab' (excludes c)
  - \( s[2:] \) is 'c d'

- Called “string slicing”

\( s = 'Hello all' \)

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>e</td>
<td>l</td>
<td>l</td>
<td>o</td>
<td></td>
<td></td>
<td>a</td>
<td>l</td>
</tr>
</tbody>
</table>

- What is \( s[:4] \)?
  - A: 'o all'
  - B: 'Hello'
  - C: 'Hell' **CORRECT**
  - D: Error!
  - E: I do not know
Other Things We Can Do With Strings

- **Operation** `in`: $s_1$ in $s_2$
  - Tests if $s_1$ “a part of” $s_2$
  - Say $s_1$ a *substring* of $s_2$
  - Evaluates to a bool

- **Examples:**
  - $s = 'abracadabra'$
  - 'a' in $s$ == True
  - 'cad' in $s$ == True
  - 'foo' in $s$ == False

- **Function** `len`: `len(s)`
  - Value is # of chars in $s$
  - Evaluates to an int

- **Examples:**
  - $s = 'abracadabra'$
  - `len(s)` == 11
  - `len(s[1:5])` == 4
  - `s[1:len(s)-1]` == 'bracadabr'
Function Calls

• Python supports expressions with math-like functions
  ▪ A function in an expression is a function call
  ▪ Will explain the meaning of this later

• Function expressions have the form fun(x,y,...)

• Examples (math functions that work in Python):
  ▪ round(2.34)
  ▪ max(a+3,24)
Function Calls

- Python supports expressions with math-like functions
  - A function in an expression is a **function call**
  - Will explain the meaning of this later
- Function expressions have the form `fun(x,y,...)`

- **Examples** (math functions that work in Python):
  - `round(2.34)`
  - `max(a+3,24)`

Arguments can be any expression.
Built-In Functions

• You have seen many functions already
  ▪ Type casting functions: `int()`, `float()`, `bool()`
  ▪ Dynamically type an expression: `type()`
  ▪ Help function: `help()`
• Getting user input: `raw_input()`
• `print <string>` is **not** a function call
  ▪ It is simply a statement (like assignment)
  ▪ But it is in Python 3.x: `print(<string>)`
Method: A Special Type of Function

- Methods are unique (right now) to strings
- Like a function call with a “string in front”
  - Usage: `string.method(x,y...)`
  - The string is an *implicit argument*
- Example: `upper()`
  - s = 'Hello World'
  - s.upper() == 'HELLO WORLD'
  - s[1:5].upper() == 'ELLO'
  - 'abc'.upper() == 'ABC'

Will see why we do it this way later in course
Examples of String Methods

- \texttt{s}_1.\texttt{index(s}_2)\\  \hspace{1cm} \text{Position of the first instance of } s_2 \text{ in } s_1

- \texttt{s}_1.\texttt{count(s}_2)\\  \hspace{1cm} \text{Number of times } s_2 \text{ appears inside of } s_1

- \texttt{s}.\texttt{strip()}\\  \hspace{1cm} \text{A copy of } s \text{ with white-space removed at ends}

- \texttt{s} = 'abracadabra'\\  \hspace{1cm} \texttt{s.index('a')} == 0

- \texttt{s}.\texttt{index('rac')} == 2

- \texttt{s}.\texttt{count('a')} == 5

- 'a b'.\texttt{strip()} == 'a b'

See Python Docs for more
Built-in Functions vs Modules

• The number of built-in functions is small
  ▪ [http://docs.python.org/2/library/functions.html](http://docs.python.org/2/library/functions.html)

• Missing a lot of functions you would expect
  ▪ **Example**: cos(), sqrt()

• **Module**: file that contains Python code
  ▪ A way for Python to provide optional functions
  ▪ To access a module, the `import` command
  ▪ Access the functions using module as a *prefix*
Example: Module math

```python
>>> import math
>>> math.cos(0)
1.0
>>> cos(0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'cos' is not defined
>>> math.pi
3.141592653589793
>>> math.cos(math.pi)
-1.0
```
Example: Module math

```python
>>> import math
>>> math.cos(0)
1.0
>>> cos(0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'cos' is not defined
>>> math.pi
3.141592653589793
>>> math.cos(math.pi)
-1.0
```
Example: Module math

```python
>>> import math
>>> math.cos(0)
1.0
>>> cos(0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'cos' is not defined
>>> math.pi
3.141592653589793
>>> math.cos(math.pi)
-1.0
```
Example: Module math

```python
>>> import math
>>> math.cos(0)
1.0
>>> cos(0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'cos' is not defined
>>> math.pi
3.141592653589793
>>> math.cos(math.pi)
-1.0
```

To access math functions

- Functions require math prefix!

Module has variables too!
Example: Module math

```python
>>> import math
>>> math.cos(0)
1.0
>>> cos(0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'cos' is not defined
>>> math.pi
3.141592653589793
>>> math.cos(math.pi)
-1.0
```

Other Modules

- **io**
  - Read/write from files
- **random**
  - Generate random numbers
  - Can pick any distribution
- **string**
  - Useful string functions
- **sys**
  - Information about your OS

Module has variables too!

Functions require math prefix!
Reading the Python Documentation

math.ceil(x)

Return the ceiling of x as a float, the smallest integer value greater than or equal to x.

http://docs.python.org/library
Function name

Module

Possible arguments

What the function evaluates to

http://docs.python.org/library
Using the `from` Keyword

```python
>>> import math
>>> math.pi
3.141592653589793
>>> from math import pi
>>> pi
3.141592653589793
>>> from math import *
>>> cos(pi)
-1.0
```

- **Be careful using from!**
  - Must prefix with module name
- **Using import is safer**
  - Modules might conflict (functions with the same name)
  - What if import both?
- **Example**: Turtles
  - Used in Assignment 4
  - 2 modules: turtle, tkturtle
  - Both have func. Turtle()
A String Puzzle (Extraction Practice)

• **Given**: a string with a parenthesis pair inside
  
  \[ s = 'labs are (usually) every week' \]

• **Goal**: expression for substring inside parentheses
  
  ▪ **Step 1**: Find the open parenthesis
    
    \[ \text{start} = s.index('(') \]
  
  ▪ **Step 2**: Store part of string **after** parenthesis in **tail**
    
    \[ \text{tail} = s[\text{start}+1:] \]
  
  ▪ **Step 3**: Get the part of the tail **before** close parenthesis
    
    \[ \text{tail}[:\text{tail.index('')}] \]
• **Given**: A string that is a list of words separated by commas, and spaces in between each comma:

```
pets = 'cat, dog, mouse, lion'
```

• **Goal**: Want second element with no spaces or commas. Put result inside of variable `answer`

Where, in the following sequence of commands, is there a (conceptual) error that prevents our goal?

A: `startcomma = info.index(',')`
B: `tail = info[startcomma+1:]`
C: `endcomma = tail.index(',')`
D: `df = tail[:endcomma]`
E: this sequence achieves the goal
• **Given:** A string that is a list of words separated by commas, and spaces in between each comma:
  
  ```python
  pets = 'cat, dog, mouse, lion'
  ```

• **Goal:** Want second element with no spaces or commas. Put result inside of variable `answer`

Where, in the following sequence of commands, is there a (conceptual) error that prevents our goal?

A: `startcomma = info.index('',')`

B: `tail = info[startcomma+1:] + 2 instead, or use`

C: `endcomma = tail.index('',')`

D: `df = tail[:endcomma] tail[:endcomma].strip()`

E: This sequence achieves the goal