Lecture 2

Strings, Functions, & Modules
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>\</td>
<td>single quote</td>
</tr>
<tr>
<td>&quot;</td>
<td>double 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>

Type: `str`
String are Indexed

• \( s = 'abc d' \)

\[
\begin{array}{|c|c|c|c|c|}
\hline
& 0 & 1 & 2 & 3 & 4 \\
\hline
a & b & c & d \\
\hline
\end{array}
\]

• 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”

• \( s = 'Hello all' \)

\[
\begin{array}{|c|c|c|c|c|c|c|c|c|}
\hline
& 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\hline
H & e & l & l & o & & a & l & l \\
\hline
\end{array}
\]

• What is \( s[3:6] \)?

\[
\begin{array}{|c|}
\hline
A: 'lo a' \\
B: 'lo' \\
C: 'lo ' \\
D: 'o ' \\
E: I do not know \\
\hline
\end{array}
\]
String are Indexed

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

- 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”

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

- What is \( s[3:6] \)?
  
  A: 'lo a'
  B: 'lo'
  C: 'lo' \hspace{1cm} \text{CORRECT}
  D: 'o '
  E: I do not know
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[:4]?
  A: 'o all'
  B: 'Hello'
  C: 'Hell'
  D: Error!
  E: I do not know
String are Indexed

- \( s = 'abc d' \)

  - 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”

- \( s = 'Hello all' \)

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

9/11/15

Strings and Modules
Other Things We Can Do With Strings

- **Operation**: `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)`

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

- `s_1.index(s_2)`
  - Position of the first instance of `s_2` in `s_1`

- `s_1.count(s_2)`
  - Number of times `s_2` appears inside of `s_1`

- `s.strip()`
  - A copy of `s` with white-space removed at ends

- `s = 'abracadabra'`

- `s.index('a') == 0`
- `s.index('rac') == 2`
- `s.count('a') == 5`

- `' a b '.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

>>> 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!
To access math functions
Reading the Python Documentation

Function name

Possible arguments

Module

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`!
- Namespaces are *safer*
  - Modules might conflict (functions w/ same name)
  - What if import both?
- **Example**: Turtles
  - Used in CS 1110
  - 2 modules: turtle, tkturtle
  - Both have func. Turtle()
How Do We Make Our Own Modules?

• Modules provide extra functions, variables
  ▪ **Example**: math provides math.cos(), math.pi

• We might want to make our own
  ▪ Custom scientific functions
  ▪ Specialized scientific constants

• This requires **two different** programs
  ▪ **Komodo Edit** to *make* a module
  ▪ **Python** to *use* the module
Python Shell vs. Modules

- Launch in command line
- Type each line separately
- Python executes as you type

- Write in a text editor
  - We use Komodo Edit
  - But anything will work
- Run module with `import`
Creating a Module

Module Contents

# module.py

""" This is a simple module. It shows how modules work"""

x = 1 + 2
x = 3 * x
x

Single line comment
(not executed)

Docstring (note the Triple Quotes)
Acts as a multiple-line comment
Useful for code documentation

x = 1 + 2
x = 3 * x
x

Commands
Executed on import

Not a command. import ignores this
# Creating a Module

## Module Contents

```python
# module.py

""" This is a simple module. 
It shows how modules work""

x = 1+2
x = 3*x
```

## Python Shell

```python
>>> import module

```

Traceback (most recent call last):

File "<stdin>", line 1, in <module>
NameError: name 'x' is not defined

```python
>>> x

9

>>> module.x

9

>>> help(module)
```

"""Module data"" must be prefixed by module name

"""Module data"" must be prefixed by module name

Prints docstring and module contents

Strings and Modules

9/11/15
Modules Must be in Working Directory!

Module you want is in this folder
Modules Must be in Working Directory!

Have to navigate to folder BEFORE running Python.

Module you want is in this folder.