**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:
      - \' single quote
      - \" double quote
      - \n new line
      - \t tab
      - \\ backslash

**String are Indexed**

- **s = 'abc d'**
  - s[0] is 'a'
  - s[4] is 'd'
  - s[5] causes an error
  - s[0:2] is 'ab'
  - s[2:] is 'c d'

  • Called “string slicing”

- **s = 'Hello all'**
  - What is s[3:6]?

<table>
<thead>
<tr>
<th>Type</th>
<th>Char</th>
<th>Meaning</th>
</tr>
</thead>
</table>
| str      | \   | backslash
|          | \t  | tab     |
|          | \n  | new line |
|          | \" | double quote |
|          | \'  | single quote |

**Other Things We Can Do With Strings**

- **Operation in** $a_1$ in $a_2$
  - Tests if $a_1$ “a part of” $a_2$
  - Say $a_1$ a **substring** of $a_2$
  - Evaluates to a bool

- **Examples**:
  - s = 'abracadabra'
  - 'a' in s == True
  - 'bad' in s == True
  - 'too' in s == False

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

- **Examples**:
  - s = 'abracadabra'
  - len(s) == 11
  - len(a[1:8]) == 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)

**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>)

- **Function calls**
  - Arguments go in (), but name() refers to function in general

- **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()
      - g = 'Hello World'
      - g.upper() == 'HELLO WORLD'
      - g[1:].upper() == 'ELLO'
      - 'abc'.upper() == 'ABC'

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

- `s1.index(s2)`
  - Position of the first instance of `s2` in `s1`

- `s1.count(s2)`
  - Number of times `s2` appears inside of `s1`

- `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

```python
>>> import math
>>> math.cos(0)
1.0
>>> math.cos(math.pi)
-1.0
```

Reading the Python Documentation

To access math functions

Functions require math prefix!

Module has variables too!

Possible arguments

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

http://docs.python.org/library

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

```python
# module.py

# This is a simple module. It shows how modules work
```

Docstring (note the Triple Quotes)

Acts as a multiple-line comment

Useful for code documentation

Commands

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

Not a command. `import` ignores this