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>

String are Indexed

- Type: str

<table>
<thead>
<tr>
<th>s</th>
<th>Type: str</th>
</tr>
</thead>
<tbody>
<tr>
<td>'abc 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”

Other Things We Can Do With Strings

- Operation in s
  - Tests if s1 "a part of" s2
  - Say s1 a substring of s2
  - 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)

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,...)
  - The string is an implicit argument
  - Example: upper()
    - s = 'Hello World'
    - s.upper() == 'HELLO WORLD'
    - s[1:5].upper() == 'ELLO'
    - 'abc'.upper() == 'ABC'

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>)

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

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

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

- Function name
- Possible arguments:
  - Return the cosine of `x` as a float, the smallest integer value greater than or equal to `x`
- Docstring (note the Triple Quotes)
  - Acts as a multiple-line comment
  - Useful for code documentation

Python Shell vs. Modules

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

Creating a Module

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

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

Module Contents

- Single line comment (not executed)
- Docstring (note the Triple Quotes)
- Acts as a multiple-line comment
- Useful for code documentation
- Commands
  - Executed on import
  - Not a command
  - `import` ignores this