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

<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>&quot;</td>
<td>backslash</td>
</tr>
</tbody>
</table>

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”

Other Things We Can Do With Strings

- Operation in: s1 in s2
  - Tests if s1 is a part of” s2
  - Say s1 is a substring of s2
  - Evaluates to a bool
- Function len: len(s)
  - Value is # of chars in s
  - Evaluates to an int

<table>
<thead>
<tr>
<th>Examples:</th>
</tr>
</thead>
<tbody>
<tr>
<td>s = 'abracadabra'</td>
</tr>
<tr>
<td>'a' in s == True</td>
</tr>
<tr>
<td>'cad' in s =&gt; True</td>
</tr>
<tr>
<td>'foo' in s =&gt; False</td>
</tr>
<tr>
<td>s[1:len(s)-1] =&gt; 'bracadabra'</td>
</tr>
</tbody>
</table>

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(2.54)`
  - `max(a, 0.64)`

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'`

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

Will see why we do it this way later in course

Examples of String Methods

• `s = 'abraaodabra'`
  • `s.index('a') == 0`
  • `s.index('ae') == 2`
• `s.count('a') == 5`
  • `' ab'.strip() == 'ab'`

See Python Docs for more

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

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.pi
3.141592653589793
```

```python
>>> from math import pi
>>> pi
3.141592653589793
```

```python
>>> from math import *
>>> 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

Using the `from` Keyword

```python
>>> import math
>>> math.pi
3.141592653589793
```

```python
>>> from math import pi
>>> pi
3.141592653589793
```

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

• Be careful using `from`!
• Using `import` is safer
• Modules might conflict (functions w/ same name)
  • What if import both?
• Example: Turtles
  • Older version of A4
  • 2 modules: `turtle`, `tkturtle`
  • Both have func `, Turtle()`