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 ' " '
- 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

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

- \( s = 'Hello all' \)
  - What is \( s[:4] \)?

<table>
<thead>
<tr>
<th>Operation</th>
<th>in</th>
<th># of chars in string</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A )</td>
<td>'o all'</td>
<td>True</td>
</tr>
<tr>
<td>( B )</td>
<td>'Hello'</td>
<td>True</td>
</tr>
<tr>
<td>( C )</td>
<td>'Hell'</td>
<td>False</td>
</tr>
<tr>
<td>( D )</td>
<td>Error</td>
<td>False</td>
</tr>
<tr>
<td>( E )</td>
<td>I do not know</td>
<td>False</td>
</tr>
</tbody>
</table>

Other Things We Can Do With Strings

- \( s = 'abc d' \)
- \( s = 'Hello all' \)
- \( \text{Operation in: } s_1 \text{ in } s_2 \)
  - Tests if \( s_1 \) a part of " \( s_2 \)
  - Say \( s_1 \) a substring of \( s_2 \)
  - Evaluates to a bool
- \( \text{Function len: } \text{len}(s) \)
  - Value is \# of chars in \( s \)
  - Evaluates to an int
- \( \text{Examples:} \)
  - \( s = 'abraodabra' \)
  - \( 'a' \) in \( s \) == True
  - \( 'bad' \) in \( s \) == True
  - \( 'foo' \) in \( s \) == False
- \( \text{Examples:} \)
  - \( s = 'abraodabra' \)
  - \( \text{len}(a) == 11 \)
  - \( \text{len}(a[1:9]) == 4 \)
  - \( \text{len}([i:len(i)-1]) == 'brasadabra' \)

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 \( \text{fun}(x,y,\ldots) \)
- \( \text{Examples} \) (math functions that work in Python):
  - \( \text{round}(2.54) \)
  - \( \text{max}(a=0.24) \)
  - Arguments can be any expression

Built-In Functions

- You have seen many functions already
  - Type casting functions: \( \text{int()}, \text{float()}, \text{bool()} \)
  - Dynamically type an expression: \( \text{type()} \)
  - Help function: \( \text{help()} \)
- Getting user input: \( \text{raw_input()} \)
- \( \text{print} <\text{string}> \) is not a function call
  - It is simply a statement (like assignment)
  - But it is in Python 3.x: \( \text{print}(\text{<string>}) \)
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'

Will see why we do it this way later in course

Examples of String Methods

- s.index(s)
  - Position of the first instance of s in s
- s.count(s)
  - Number of times s appears inside of s
- s.strip()
  - A copy of s with whitespace removed at ends

Example: Module math

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

Example: Turtles

- Older version of A4
- 2 modules: turtle, tkturtle
- Both have func. Turtle()