Strings are Indexed

- `s = 'abc d'`
- `s = 'Hello all'`

<table>
<thead>
<tr>
<th>Access characters with []</th>
</tr>
</thead>
<tbody>
<tr>
<td>s[0] is 'a'</td>
</tr>
<tr>
<td>s[4] is 'd'</td>
</tr>
<tr>
<td>s[5] causes an error</td>
</tr>
<tr>
<td>s[0:2] is 'ab' (excludes c)</td>
</tr>
<tr>
<td>s[2:] is 'c d'</td>
</tr>
<tr>
<td>Called “string slicing”</td>
</tr>
</tbody>
</table>

- What is s[3:6]?

Other Things We Can Do With Strings

- **Operation in:** `s1 in s2`
  - 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'`

Defining a String Function

```python
def middle(text):
    """Returns: middle 3rd of text
    Param text: a string"""
    # Get length of text
    size = len(text)
    # Start of middle third
    start = size/3
    # End of middle third
    end = 2*size/3
    # Get the text
    result = text[start:end]
    # Return the result
    return result
```

Not All Functions Need a Return

```python
def greet(n):
    """Prints a greeting to the name n
    Parameter n: name to greet
    Precondition: n is a string""
    print 'Hello '+n+'!
    print 'How are you?'
```

Procedures vs. Fruitful Functions

<table>
<thead>
<tr>
<th>Procedures</th>
<th>Fruitful Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functions that <strong>do something</strong></td>
<td>Functions that give a <strong>value</strong></td>
</tr>
<tr>
<td>Call them as a <strong>statement</strong></td>
<td>Call them in an <strong>expression</strong></td>
</tr>
<tr>
<td>Example: <code>greet('Walker')</code></td>
<td>Example: <code>x = round(2.56,1)</code></td>
</tr>
</tbody>
</table>

Print vs. Return

<table>
<thead>
<tr>
<th>Print</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>Displays a value on screen</td>
<td>Defines a function’s value</td>
</tr>
<tr>
<td>* Used primarily for testing</td>
<td>* Important for calculations</td>
</tr>
<tr>
<td>* Not useful for calculations</td>
<td>* But does not display anything</td>
</tr>
</tbody>
</table>

```python
def print_plus(n):
    print (n+1)
>>> x = print_plus(2)
3
```

```python
def return_plus(n):
    return (n+1)
>>> x = return_plus(2)
3
```
Advanced String Features: Method Calls

- Methods calls are unique (right now) to strings
- Like a function call with a “string in front”
  - Usage: \texttt{string.method(x, y, \ldots)}
  - 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

- \texttt{s.s.index(s2)}
  - Position of the first instance of \texttt{s2} in \texttt{s1}
- \texttt{s.s.count(s2)}
  - Number of times \texttt{s2} appears inside of \texttt{s1}
- \texttt{s.strip()}
  - A copy of \texttt{s} with white-space removed at ends

String Extraction Example

```python
def firstparens(text):
    """Returns: substring in ()
    Uses the first set of parens
    Param text: a string with ()"
    # Find the open parenthesis
    start = text.index('(')
    # Store part AFTER paren
tail = text[start+1:]
    # Find the close parenthesis
    end = tail.index(')')
    # Return the result
    return tail[:end]
```

```python
>>> s = 'One (Two) Three'
>>> firstparens(s)
'Two'
```

String Extraction Puzzle

```python
def second(thelist):
    """Returns: second in the list
    The list is a sequence of words separated by commas, spaces.
    Ex: second('A, B, C') => 'B'
    Param thelist: a list of words"
    start = thelist.index(',')
    tail = thelist[start+1:]
    end = tail.index(',')
    result = tail[:end]
    return result
```

```python
>>> second('cat, dog, mouse, lion')
'dog'
```

See Python Docs for more