**COMP 141**

CS1: Programming Fundamentals  
October 24, 2014

---

**Announcements**

Reminders:
- MPL Assignment 5 – due 10/28
- Program 5 - due 10/30

---

**Using len function**

# Prints 1 letter of city on each line

city = 'Boston'
index = 0
while index < len(city):
    print(city[index])
    index += 1

# Equivalent Code

city = 'Boston'
for index in range(0, len(city)):
    print(city[index])

---

**String Testing Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>isalpha()</td>
<td>Returns true if the string contains only alphabetic letters or digits and is</td>
</tr>
<tr>
<td></td>
<td>at least one character in length. Returns false otherwise.</td>
</tr>
<tr>
<td>isalnum()</td>
<td>Returns true if the string contains only alphabetic letters, and is at least</td>
</tr>
<tr>
<td></td>
<td>one character in length. Returns false otherwise.</td>
</tr>
<tr>
<td>isdigit()</td>
<td>Returns true if the string contains only numeric digits and is at least one</td>
</tr>
<tr>
<td></td>
<td>character in length. Returns false otherwise.</td>
</tr>
<tr>
<td>islower()</td>
<td>Returns true if all of the alphabetic letters in the string are lowercase,</td>
</tr>
<tr>
<td></td>
<td>and the string contains at least one alphabetic letter. Returns false</td>
</tr>
<tr>
<td></td>
<td>otherwise.</td>
</tr>
<tr>
<td>isspace()</td>
<td>Returns true if the string contains only whitespace characters, and is at</td>
</tr>
<tr>
<td></td>
<td>least one character in length. Returns false otherwise. (Whitespace</td>
</tr>
<tr>
<td></td>
<td>characters are spaces, newlines (\n), and tabs (\t).</td>
</tr>
<tr>
<td>isupper()</td>
<td>Returns true if all of the alphabetic letters in the string are uppercase,</td>
</tr>
<tr>
<td></td>
<td>and the string contains at least one alphabetic letter. Returns false</td>
</tr>
<tr>
<td></td>
<td>otherwise.</td>
</tr>
</tbody>
</table>
Example using isupper()

```python
Example using isupper()

def main():
    # Create a variable to use to hold the count.
    count = 0
    # Get a string from the user.
    my_string = input('Enter a sentence: ')
    # Count the uppercase letters
    for ch in my_string:
        if ch.isupper():
            count += 1
    # Print the result.
    print(count, 'of the letters were uppercase.')
    # Call the main function.
    main()
```

String Modification Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lower()</td>
<td>Returns a copy of the string with all alphabetic letters converted to lowercase. Any character that is already lowercase, or is not an alphabetic letter, is unchanged.</td>
</tr>
<tr>
<td>strip()</td>
<td>Returns a copy of the string with all leading whitespace characters removed. Trailing whitespace characters are spaces, newlines (\n), and tabs (\t) that appear before the beginning of the string.</td>
</tr>
<tr>
<td>strip()</td>
<td>Returns a copy of the string with all trailing whitespace characters removed. Trailing whitespace characters are spaces, newlines (\n), and tabs (\t) that appear after the end of the string.</td>
</tr>
<tr>
<td>strip(char)</td>
<td>The clear argument is a string containing a character. Return a copy of the string with all instances of char that appear at the beginning of the string removed.</td>
</tr>
<tr>
<td>strip()</td>
<td>Returns a copy of the string with all leading and trailing whitespace characters removed.</td>
</tr>
<tr>
<td>strip(char)</td>
<td>The clear argument is a string containing a character. The method returns a copy of the string with all instances of char that appear at the end of the string removed.</td>
</tr>
<tr>
<td>strip()</td>
<td>Returns a copy of the string with all instances of char that appear at the beginning and the end of the string removed.</td>
</tr>
<tr>
<td>strip(char)</td>
<td>Returns a copy of the string with all instances of char that appear at the beginning and the end of the string removed.</td>
</tr>
<tr>
<td>strip()</td>
<td>Returns a copy of the string with all instances of char that appear at the end of the string removed.</td>
</tr>
</tbody>
</table>

Stripping Newline from a String

```python
Example

weapon = input("Enter rock(R), paper(P), or scissors(S)")
weapon = weapon.lower()
if weapon == 'r' or weapon == 'p' or weapon == 's':
    validMove = True
else:
    validMove = False
```

Program Output

<table>
<thead>
<tr>
<th>Program Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Locke</td>
</tr>
<tr>
<td>David Hume</td>
</tr>
<tr>
<td>Edmund Burke</td>
</tr>
</tbody>
</table>
Accessing Characters Review

Strings are stored character by character. Each character in a string is numbered by its position:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;C&quot;</td>
<td>&quot;o&quot;</td>
<td>&quot;m&quot;</td>
<td>&quot;p&quot;</td>
<td>&quot;u&quot;</td>
<td>&quot;t&quot;</td>
<td>&quot;e&quot;</td>
<td>&quot;r&quot;</td>
</tr>
</tbody>
</table>

The numbers shown here above the characters are called indices (singular: index) or positions.

Negative Indices

Negative indexing can be used. Particularly useful for getting characters near the end of a string.

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>-7</td>
<td>-6</td>
<td>-5</td>
<td>-4</td>
<td>-3</td>
<td>-2</td>
<td>-1</td>
</tr>
</tbody>
</table>

\[s[2] \text{ is the same as } s[-6] \text{ both refer to "m"}

To find last letter in string use: \[s[-1]\]

String Indices

- Two ways to use square brackets
  - 1 number inside -> gives you 1 character of a string
    - \(s[0]\) gives you the first character in \(s\)
    - If \(s = \text{"Computer"}\), \(s[0]\) gives you ‘C’
  - 2 numbers inside (separated by a colon) -> gives you a substring or string slice

String Slicing

- **Slice**: span of items taken from a sequence, known as substring
  - Slicing format: \(\text{string[start : end]}\)
    - Expression will return a string containing a copy of the characters from \(\text{start}\) up to, but not including, \(\text{end}\)
    - If \(\text{start}\) not specified, 0 is used for start index
    - If \(\text{end}\) not specified, \(\text{len(string)}\) is used for end index
  - Slicing expressions can include a step value and negative indexes relative to end of string
String Slicing

\[ s[a:b] \] gives you a substring of \( s \) starting from index \( a \) and ending at index \( b-1 \).

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>'C'</td>
<td>'o'</td>
<td>'m'</td>
<td>'p'</td>
<td>'u'</td>
<td>'t'</td>
<td>'e'</td>
<td>'r'</td>
<td></td>
</tr>
</tbody>
</table>

- \( s[0:1] \) -> “C” just like \( s[0] \)
- \( s[0:2] \) -> “Co”
- \( s[0:7] \) -> “Compute”
- \( s[3:6] \) -> “put”
- \( s[0:8] \) -> “Computer”

Indices Don’t have to be Literal Numbers

Say we have this code:

```python
s = input("Type in a string: ")
x = int(len(s) / 2)
print s[0:x])
```

What does this print?

More Fun with Indices

- Examples using negative indices
- A negative index counts from the right side of the string, rather than from the left

```
s = "Computer"
print s[-1])       #prints r
print s[-3:len(s)]) #prints ter
print s[1:-1])     #prints compute
```

More Fun with Indices

- Slices don’t need both left and right indices
- Missing left -> use 0 [far left of string]
- Missing right -> use len(s) [far right of string]

```
s = "Computer"
print(s[1:])       #prints computer
print(s[:5])       #prints Compu
print(s[-2:])      #prints er
```
Practice

- Write a function called `total_seconds` that takes one string argument. This argument will be a string of the form "M:SS" where M is a number of minutes (a single digit) and SS is a number of seconds (2 digits). This function should calculate the total number of seconds in this amount of time and return it as an integer.

- Write a main function that lets the user type in a time as a string and will call your `total_seconds` function. Your output statement should be in the main function as well.

Next Time

More Strings