7. String Methods

Topics:

Using Methods from the string class

Iterating through a string with for

Data + Functions Together

“The square root of nine is three.”

The tone of this comment is that the square root function can be applied to numbers like nine.

“Three is nine’s square root.”

The tone of this comment is that the number nine (like all numbers) comes equipped with a sqrt function.

Methods

A special kind of function that is very important to object-oriented programming is called a method.

In this style of programming, there is a tight coupling between structured data and the methods that work with that data.

Three String Methods

count  How many times does string t occur in a string s?

find  Where is the first occurrence of string t in a string s?

replace  In a string s replace all occurrences of a string s1 with a string s2.

Possible Designs

count  How many times does string t occur in a string s?

A function with two parameters: n = count(t, s)
Possible Designs

**find** Where is the first occurrence of string t in a string s?

`'ITH-JFK-ITH' --> find --> 3`

A function with two parameters. ??? n = find(t, s) ???

Possible Designs

**replace** In a string s replace all occurrences of a string s1 with a string s2.

`'ITH-JFK-ITH' --> replace --> '??-JFK-??'

A function with three parameters. ??? sNew = replace(s, s1, s2) ???
count
The Formal Definition

If \(s_1\) and \(s_2\) are strings, then
\[ s_1.count(s_2) \]
returns an int value that is the number of occurrences of string \(s_2\) in string \(s_1\).

Note, in general \(s_1.count(s_2)\) is not the same as \(s_2.count(s_1)\).

Using count: An Example

```python
# Count the number of vowels...
A = 'auric goldfinger'
n = 0
n = n + A.count('a')
n = n + A.count('e')
n = n + A.count('i')
n = n + A.count('o')
n = n + A.count('u')
print n
```

Illegal: \(n = A.count('a' or 'e' or 'i' or 'o' or 'u')\)

String Methods: find

```python
>>> s = 'ITH-JFK-ITH'
>>> idx = s.find('JFK')
```

\[
\begin{array}{cccccccccccc}
\text{s} & \rightarrow & I & T & H & - & J & F & K & - & I & T & H \\
& 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
\text{idx} & \rightarrow & 4 \\
\end{array}
\]

\(s_1.index(s_2)\) the index of the first occurrence of string \(s_2\) in string \(s_1\)

```python
>>> s = 'ITH-JFK-ITH'
>>> idx = s.find('RFK')
```

\[
\begin{array}{cccccccccccc}
\text{s} & \rightarrow & I & T & H & - & J & F & K & - & I & T & H \\
& 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
\text{idx} & \rightarrow & -1 \\
\end{array}
\]

\(s_1.index(s_2)\) evaluates to \(-1\) if there is no occurrence of \(s_2\) in \(s_1\)

find
The Formal Definition

If \(s_1\) and \(s_2\) are strings, then
\[ s_1.find(s_2) \]
returns an int value that is the index of the first occurrence of string \(s_2\) in string \(s_1\).

If there is no such occurrence, then the value \(-1\) is returned.

Using find : Some Examples

```python
s = 'nine one one'
n1 = s.find('one')
n2 = s.find('two')
n3 = s.find('nine')
```

\(n1 \rightarrow 5\)  \(n2 \rightarrow -1\)  \(n3 \rightarrow -1\)
The replace Method

\[ s = \text{'one hundred and one'} \]
\[ t = s.\text{replace(' ','-')} \]

- \( s \rightarrow \text{'one hundred and one'} \)
- \( t \rightarrow \text{'one-hundred-and-one'} \)

Replacing one character with another

The null string has length 0.

The replace Method

\[ s = \text{'one hundred and one'} \]
\[ t = s.\text{replace('','')} \]

- \( s \rightarrow \text{'one hundred and one'} \)
- \( t \rightarrow \text{'onehundredandone'} \)

Replacing each blank with the null string

The replace Method

\[ s = \text{'one hundred and one'} \]
\[ t = s.\text{replace('x','')} \]

- \( s \rightarrow \text{'one hundred and one'} \)
- \( t \rightarrow \text{'one hundred and one'} \)

No change if the character to be replaced is missing

The replace Method

\[ s = \text{'one hundred and one'} \]
\[ t = s.\text{replace('two','seven')} \]

- \( s \rightarrow \text{'one hundred and one'} \)
- \( t \rightarrow \text{'seven hundred and seven'} \)

Replacing one substring with another

The replace Method

\[ s = \text{'one hundred and one'} \]
\[ t = s.\text{replace('two','seven')} \]

- \( s \rightarrow \text{'one hundred and one'} \)
- \( t \rightarrow \text{'one hundred and one'} \)

No change if the designated substring is missing

replace

The Formal Definition

If \( s, s1 \) and \( s2 \) are strings, then

\[ s.\text{replace}(s1,s2) \]

returns a copy of the string \( s \) in which every non-overlapping occurrence of the string \( s1 \) is replaced by the string \( s2 \).

If \( s1 \) is not a substring of \( s \), then the returned string is just a copy of \( s \).
Using `replace`: Some Examples

```python
s = 'xxx'
t1 = s.replace('x', 'o')
t2 = s.replace('xx', 'o')
t3 = s.replace('xx', 'oo')
```

<table>
<thead>
<tr>
<th>t1</th>
<th>t2</th>
<th>t3</th>
</tr>
</thead>
<tbody>
<tr>
<td>'ooo'</td>
<td>'ox'</td>
<td>'oox'</td>
</tr>
</tbody>
</table>

Replace does Not Replace

- `s.replace(s1, s2)` does not change the value of `s`.
- It produces a copy of `s` with the specified replacements.
- You are allowed to overwrite the "original" `s` with the its "updated" copy:
  ```python
  s = s.replace(s1, s2)
  ```

Illegal!

```python
s = 'abcdefg'  # Strings are immutable. They cannot be changed.
s[5] = 'x'
```

Have to "live with" the `replace` function, slicing, and concatenation

```python
s = 'abcdefg'
S = s[:5] + 'x' + s[6:]
```

Upper and Lower Methods

```python
s = 'A2sh?'  # s -> 'A2sh?'
t1 = s.upper()  # t1 -> 'A2SH?'
t2 = s.lower()  # t2 -> 'a2sh?'
```

Boolean-Valued Methods

These methods return either True or False:

- `islower()`
- `isupper()`
- `isalnum()`
- `isalpha()`
- `isdigit()`

<table>
<thead>
<tr>
<th>s='ab3?'</th>
<th>s='AbcD'</th>
<th>s='AB3'</th>
</tr>
</thead>
<tbody>
<tr>
<td>s.islower()</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>s.isupper()</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>
### Boolean-Valued Methods

<table>
<thead>
<tr>
<th></th>
<th>'23'</th>
<th>'5a7'</th>
<th>'ab'</th>
<th>'-2.3'</th>
</tr>
</thead>
<tbody>
<tr>
<td>s.isalnum()</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>s.isalpha()</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>s.isdigit()</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>

### Useful String Constants

- **alpha** = `string.letters`
  ```python
  abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
  ```
- **specialChar** = `string.punctuation`
  ```python
  !"#$%&'()*+,./:;<=>?@[\]^_`{|}~
  ```
- **TheDigits** = `string.digits`
  ```python
  01234567890
  ```

### The “Dot” Notation—Again

We have seen it with modules and `import`:

- `math.sqrt`
- `math.pi`

The “folder” metaphor.

The “dot” means “go inside and get this”:

- `math.pi = 3.1416`
- `math.sqrt`

### String is a “Special” Module

The “folder” metaphor.

The “dot” means “go inside and get this”:

- `string` is actually a “class”. More in a few lectures.
Iterating Through a String

Two problems we cannot easily solve:

1. Given a string $s$, assign to $t$ the "reversed" string. $'abcd' \rightarrow 'dcba'$

2. Given a string $s$, how many digit characters does it contain? $'1or2or3' \rightarrow 3$

The Reverse String Problem

$$s = 'abcd' \quad t = ''$$
$$\text{for } c \text{ in } s:$$
$$t = c + t$$
$$t \rightarrow 'dcba'$$

How does the for loop work?

The Number-of-Digits Problem

$$s = '2x78y'$$
$$n = 0$$
$$\text{for } c \text{ in } s:$$
$$\text{if } c \text{.isdigit}():$$
$$n = n + 1$$
$$n \rightarrow 3$$

How does the for loop work?

Using for to Traverse a String Character-by-Character

$$s = 'abcd'$$
$$\text{for } c \text{ in } s:$$
$$\text{print } c$$
$$\text{Output: } a \quad b \quad c \quad d$$

In this example, the "for-loop" variable is $c$. One at a time, it takes on the value of each character in $s$.

The Reverse String Problem

$$s = 'abcd'$$
$$t = ''$$
$$\text{for } c \text{ in } s:$$
$$\text{print } t$$
$$c \rightarrow 'a'$$

At the start of the loop, $c$ is assigned the zeroth character in $s$. 

The loop body is executed using that value in $c$. 

The Reverse String Problem

$$s = 'abcd'$$
$$t = ''$$
$$\text{for } c \text{ in } s:$$
$$\text{print } t$$
$$c \rightarrow 'a'$$

The loop body is executed using that value in $c$. 

The Reverse String Problem

```python
s = 'abcd'
t = ''
for c in s:
    t = c + t
print t
```

The loop body is executed using that value in `c`.

The next time through the loop, `c` is assigned the first character in `s`.

The Reverse String Problem

```python
s = 'abcd'
t = ''
for c in s:
    t = c + t
print t
```

The loop body is executed using that value in `c`.

The next time through the loop, `c` is assigned the second character in `s`.

The Reverse String Problem

```python
s = 'abcd'
t = ''
for c in s:
    t = c + t
print t
```

The loop body is executed using that value in `c`.

The next time through the loop, `c` is assigned the third character in `s`.
The Reverse String Problem

```python
s = 'abcd'
t = ''
for c in s:
    t = c + t
print t
```

The loop body is executed using that value in `c`.

The string has been traversed. The iteration ends. The next statement after the loop is executed. Indentation important.

```python
for <loop variable> in <string>:
    Loop Body
```

If the string has length `n`, then the loop body is executed `n` times.
**for-loop Mechanics**

for x in y:

    Loop Body

Let x = y[0] and then execute the loop body.
Let x = y[1] and then execute the loop body.
Let x = y[2] and then execute the loop body.
   etc
Let x = y[n-1] and then execute the loop body.

**The Number-of-Digits Problem**

Given a string s, how many of its characters are digit characters?

'a10b20c30d40' \(\rightarrow\) 8

**The Number-of-Digits Problem**

s = '2z78y'
n = 0
for x in s:
    if x.isdigit():
        n = n + 1
    print n

At the start of the loop, x is assigned the zeroth character in s.

**The Number-of-Digits Problem**

s = '2z78y'
n = 0
for x in s:
    if x.isdigit():
        n = n + 1
        print n

The loop body is executed using that value in x.

**The Number-of-Digits Problem**

s = '2z78y'
n = 0
for x in s:
    if x.isdigit():
        n = n + 1
        print n

The next time through the loop, x is assigned the first character in s.
The Number-of-Digits Problem

s = '2z78y'
for x in s:
    if x.isdigit():
        n = n+1
    print n
x = 'z'

The loop body is executed using that value in x.

The Number-of-Digits Problem

s = '2z78y'
for x in s:
    if x.isdigit():
        n = n+1
    print n
x = '8'

The next time through the loop, x is assigned the third character in s.
The Number-of-Digits Problem

\[
\begin{align*}
\text{s} &= '2z78y' \\
\text{n} &= 0 \\
\text{for } x \text{ in } s: \\
    &\quad \text{if } x.\text{isdigit}(): \\
    &\quad \quad n = n + 1 \\
\text{print n}
\end{align*}
\]

The loop body is executed using that value in \( x \).

Function for Reversing Strings

```python
def Reverse(s):
    
    """ Returns a string that is obtained from \( s \) by reversing the order of its characters. """
    t = '' # The empty string
    for c in s:
        t = c + t # Repeated concatenation
    return t
```

Function for Counting Digits

```python
def nDigits(s):
    
    """ Returns an int whose value is the number of digit characters that are in \( s \). """
    n = 0;
    for c in s:
        # Increment n if c is a digit
        if c.\text{isdigit}():
            n = n + 1
    return n
```