Introduction to Computer Programming
Section 8
Lecture 19
Analyzing and Transforming Strings
Natural Language “Understanding”

• Everyone (retailers, government, investment banks, etc) wants to understand social media and news feeds:

  What are people talking about?
  What is latest news?
  What are the popular opinions?
  Who is a threat to national security?

• People also want to ask questions and get answers
  – Knowledge retrieval, Problem resolution
Natural Language Understanding is a Hard Problem!

- You can go a long way with simple word search within a specific knowledge domain.

Example: automated phone repair service:

**Person:** My phone doesn’t work.

**Computer:** What is your problem exactly?

**Person:** I’m not getting a dial tone.

Computer goes to dial tone script.
Text Analysis with Python

• What capabilities does Python have to help analyze and transform strings?

  – A string is not just a single data value, but a list of characters
  – We need to be able to get at the individual characters of strings
Accessing the Individual Character in a String

• To access an individual character in a string:
  – Use a `for` loop

```python
name = "Juliet"
for ch in name:
    print(ch)
```
Figure 9-1  Iterating over the string 'Juliet'

<table>
<thead>
<tr>
<th>1st Iteration</th>
<th>for ch in name: print(ch)</th>
<th>2nd Iteration</th>
<th>for ch in name: print(ch)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>name → 'Juliet'</td>
<td>name → 'Juliet'</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ch → 'J'</td>
<td>ch → 'u'</td>
<td></td>
</tr>
<tr>
<td>3rd Iteration</td>
<td>for ch in name: print(ch)</td>
<td></td>
<td>for ch in name: print(ch)</td>
</tr>
<tr>
<td></td>
<td>name → 'Juliet'</td>
<td>name → 'Juliet'</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ch → 'l'</td>
<td>ch → 'i'</td>
<td></td>
</tr>
<tr>
<td>5th Iteration</td>
<td>for ch in name: print(ch)</td>
<td></td>
<td>for ch in name: print(ch)</td>
</tr>
<tr>
<td></td>
<td>name → 'Juliet'</td>
<td>name → 'Juliet'</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ch → 'e'</td>
<td>ch → 't'</td>
<td></td>
</tr>
<tr>
<td>6th Iteration</td>
<td>for ch in name: print(ch)</td>
<td></td>
<td>for ch in name: print(ch)</td>
</tr>
</tbody>
</table>
Application: String Validation

We often need to validate that a string conforms to a particular format.

• Date: mm/dd/yyyy
• Floating point number: 52.345
• Password must conform to rules
Validation Pattern

# Assume `string` is Valid (or Invalid)

# Iterate over characters in `string`

for ch in `string`:
    # test to Invalidate (or Validate)

# Final test to Invalidate (or Validate)
Example: Valid if string has no spaces

# Assume string is Valid
valid = True

# Iterate over characters in string
for ch in string:
    # Invalidate if find a space
    if ch == " " :  
        valid = False

print(valid)
Accessing the Individual Character in a String

• To access an individual character in a string:

  – Each character has an index specifying its position in the string, starting at 0

```python
character = my_string[7]
```
Accessing the Individual Characters in a String

\[
\text{mystring} = \text{‘Roses are red’}
\]

\[
\begin{align*}
\text{mystring}[0] & \quad == \quad \text{“R”} \\
\text{mystring}[7] & \quad == \quad \text{“r”} \\
\text{mystring}[12] & \quad == \quad \text{“d”} \\
\text{mystring}[13] & \quad -> \quad \text{ERROR!!!}
\end{align*}
\]
**Accessing the Individual Characters in a String**

mystring = ‘Roses are red’

# Loop to print all characters in mystring

for i in range(13) :
    ch = mystring[i]
    print(ch)
Accessing the Individual Characters in a String

• **IndexError** runtime exception will occur if:
  – You try to use an index that is out of range for the string
    – Likely to happen when loop iterates beyond the end of the string

• `len(string)` function can be used to obtain the length of a string
  – Useful to prevent loops from iterating beyond the end of a string
Accessing the Individual Characters in a String

mystring = 'Roses are red'

# Loop to print all characters in mystring
for i in range(len(mystring)) :
    ch = mystring[i]
    print(ch)
Testing and Searching Strings

• You can use the `in` operator to determine whether one string is contained in another string

```python
if string1 in string2 :
    print("found", string1)
```

• Similarly you can use the `not in` operator to determine whether one string is not contained in another string
Modifying Strings

• Sometimes you want to create a new string from an existing string:

Examples:

• For creating a string to print
• Encoding or decoding a message
• Correcting grammar or spelling
Modifying Strings

String Concatenation: “abc” + “def” # “abcdef”

String Replication: “abc” * 3 # “abcabcabc”

String functions:

str.upper(“abc”) # “ABC”
str.lower(“ABC”) # “abc”
Strings Are Immutable

• Strings are immutable
  – Once they are created, they cannot be changed
    • Concatenation doesn’t actually change the existing string, but rather creates a new string and assigns the new string to the previously used variable
  – Cannot use an expression of the form
    
    ```
    string[index] = new_character
    ```
    • Statement of this type will raise an exception
Strings Are Immutable (cont’d.)

**Figure 9-4** The string ‘Carmen’ assigned to name

```
name = 'Carmen'

name ----> Carmen
```

**Figure 9-5** The string ‘Carmen Brown’ assigned to name

```
name = name + ' Brown'

name ----> Carmen
          |    ----> Carmen Brown
```