CSCI 141
Lecture 18
Strings: Slicing, String Methods, Comparison and in operators
Announcements

• A4 is due Friday.

• Bonus points for reviewing exams will be awarded and curved scores transferred to Canvas later this week.

• I corrected grading on a couple questions
Goals

• Know how Python interprets negative indices into strings.

• Know how to use slicing to get substrings

• Know how to use a few of the basic methods of string objects:
  • upper, lower, find, replace

• Understand the behavior of the following operators on strings:
  • <, >, ==, !={}, in, and not in
  • Understand the meaning of lexicographic ordering

• Understand the meaning and implications of strings being immutable objects.
Last time...

- Review what we know already about strings:
  - the str type, + and * operators, len function

```python
type("hello")

print("Hello")

"Hello" + "World"

len("abc")

"na" * 16 + " Batman!"
```
Last time…

• Know how to iterate over tuples and **strings** using for loops

```python
def remove_vowels(string):
    """ Print string, but with no vowels. Don't
count y as a vowel.
    Pre: no upper case vowels.
    """
    result = ""
    for letter in string:
        # letter has the current letter in the string
        if not (letter == "a" or letter == "e" or letter == "i" \
                or letter == "o" or letter == "u"):
            result = result + letter
    return result
```
Last time...

- Know how to **index** into a string

```python
outlook = "Summer is near"
outlook[0]  # => "S"
outlook[4]  # => "e"
outlook[6]  # => " \\
```

*Indices in Python begin at 0.*

*Spaces are characters too!*

```python
outlook[6]  # => " \\
```
Indexing into Strings

**Index:** 0 1 2 3 4 5 6 7 8 9 10 11 12 13

**Value:** Summer is near

**ABCD:** What is the index of the last character of a string $s$?

A. $\text{len}(s) - 1$
B. $\text{len}(s)$
C. $\text{len}(s) + 1$
D. 42
A consequence of indexing -
Another way to loop through strings:

```python
for letter in a_string:
    print(letter, "-", sep=" ", end=" ")
```

is equivalent to

```python
for i in range(len(a_string)):
    print(a_string[i], "-", sep=" ", end=" ")
```
Nifty Python Feature: Negative Indices

Negative indices count backwards from len(s):

<table>
<thead>
<tr>
<th>Index:</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>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>S</td>
<td>u</td>
<td>m</td>
<td>m</td>
<td>e</td>
<td>r</td>
<td>i</td>
<td>s</td>
<td>n</td>
<td>e</td>
<td>a</td>
<td>r</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Index:</td>
<td>-14</td>
<td>-13</td>
<td>-12</td>
<td>-11</td>
<td>-10</td>
<td>-9</td>
<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>

Two possible ways to remember how this works:

-1 is always the last character, and indices count backwards from there.

```
a_string[-5]
```

is equivalent to

```
a_string[len(a_string)-5]
```
last_name = "wehrwein"

print(last_name[a] == last_name[b])

For which assignment of a and b does the above not print True?

A. a = 1  C. a = -8
   b = 5  b = -4

B. a = 1  D. a = -2
   b = 7  b = 6
Today’s Quiz

• 3 minutes
Today’s Quiz

• 3 minutes

• Working with a neighbor: do your answers agree? (2 minutes)
def remove_comments(string):
    """ Return a copy of string, but with all characters starting with the first # symbol removed. If there is no # in the string, return input unchanged. """

    Hint: use a while loop!

# Example:

remove_comments("a = b # assign b to a")
# => "a = b "
Slicing: indexing substrings

alph = "abcdefghij"
alph[0] # => "a"
alph[4] # => "e"

index of first character 1 + index of last character

Slicing syntax: string[start:end]

alph[0:5] # => "abcde"
alph[0:10] # => "abcdefghij"
alph[5:-2] # => "fgh"
Slicing: indexing substrings

\[
\text{alph} = "abcdefghij"
\]

<table>
<thead>
<tr>
<th>Ind</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>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Val</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e</td>
<td>f</td>
<td>g</td>
<td>h</td>
<td>i</td>
<td>j</td>
</tr>
</tbody>
</table>

Slicing syntax: \text{string}[\text{start}:\text{end}]

- index of first character
- 1 + index of last character

- If omitted, start defaults to 0
- If omitted, end defaults to \text{len(string)}

\text{alph}[:4] \quad \# \quad => \quad "abcd"
\text{alph}[5:] \quad \# \quad => \quad "fghij"
String Slicing: Exercise

last_name = "Wehrwein"

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>W</td>
<td>e</td>
<td>h</td>
<td>r</td>
<td>w</td>
<td>e</td>
<td>i</td>
<td>n</td>
</tr>
</tbody>
</table>

Which of the above evaluates to "in"?

A. last_name[7:8]
B. last_name[6:-1]
C. last_name[-3:]
D. last_name[-2:8]
Strings are objects.

We’ve seen other objects before: turtles!

Turtles had methods:

```python
# turtle module function (turtle constructor)
t = turtle.Turtle()
# method of a turtle object
iforward(100)
```

variable that refers to a turtle object
Strings are objects too - they also have methods.

Turtles had methods:

```python
last_name = "Wehrwein"
last_name.upper()
```
Strings have many methods

here are a few of them:

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>upper</td>
<td>none</td>
<td>Returns a string in all uppercase</td>
</tr>
<tr>
<td>lower</td>
<td>none</td>
<td>Returns a string in all lowercase</td>
</tr>
<tr>
<td>strip</td>
<td>none</td>
<td>Returns a string with the leading and trailing whitespace removed</td>
</tr>
<tr>
<td>count</td>
<td>item</td>
<td>Returns the number of occurrences of item</td>
</tr>
<tr>
<td>replace</td>
<td>old, new</td>
<td>Replaces all occurrences of old substring with new</td>
</tr>
<tr>
<td>find</td>
<td>item</td>
<td>Returns the leftmost index where the substring item is found, or -1 if not found</td>
</tr>
</tbody>
</table>
String methods: demo
upper, lower, count, replace, find, strip
String methods: demo
upper, lower, count, replace, find, strip

```python
word = "Banana"
word.upper()
word.lower()
word.count("a")
word.replace("a", "A")

line = " snails are out "
line.find("s")
line.find("snails")
line.find("banana")
line.strip()

phrase = "WWU is in Bellingham"
phrase = phrase[:19] + phrase[19].upper()
```
String Methods: More

The textbook (Section 9.5) has a more complete listing of string methods:
http://interactivepython.org/runestone/static/thinkcspy/Strings/StringMethods.html

The Python documentation has full details of the str type and all its methods:
https://docs.python.org/3/library/stdtypes.html#str

You should know how to use upper, lower, replace, and find.
Worksheet - Exercise 2

```
phrase = "WWU is in Bellingham"
phrase = phrase[0:19] + phrase[19].upper()

Write a function that capitalizes the last letter of any string:

def capitalize_last(in_str):
    """Return a copy of in_str with its last letter capitalized."
    """

# Example:
capitalize_last("Mix") # => "MiX"
```
Rewrite the function from Exercise 1 using the `find` method and slicing to avoid using a loop.

```python
def remove_comments(string):
    """ Return a copy of string, but with all characters starting with the first # symbol removed. If there is no # in the string, return input unchanged. """
```
Next time: Lists