Lecture Notes
CSC111
Week 4

Dominique Thiébaut
dthiebaut@smith.edu
Chapter 5 in Zelle
Logistic (lab cancelled)

Indexing in Strings

Indexing in Lists

Splitting Strings into Lists

String Methods
Logistic (lab cancelled)

**Indexing in Strings**

Indexing in Lists

Splitting Strings into Lists

String Methods
Data is kept in collections of “things”

We like to keep information in numbered boxes in memory
For example: strings
Strings are collections of characters

name = "ALIBABA"
name = "ALIBABA"

Important conceptual change in the way we look at string
name

0 1 2 3 4 5 6

ALIBABA
name

ALIBABA

name[1]
<table>
<thead>
<tr>
<th>name</th>
<th>A</th>
<th>L</th>
<th>I</th>
<th>B</th>
<th>A</th>
<th>B</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td></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>
name

\[ \begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 \\
\text{ALIBABA} & \\
-7 & -6 & -5 & -4 & -3 & -2 & -1 \\
\end{array} \]

\text{name[-6]} = 'L'
The diagram illustrates a sequence named `name`, which is represented as `ALIBABA`. The arrows indicate:

- `name[-2]` points to 'B'
- `name[-6]` points to 'L'

The positions are labeled as follows:

-0 1 2 3 4 5 6

And the numbers:

-7 -6 -5 -4 -3 -2 -1
There are two different ways to access the last character of a string. Which are they?
Demo Time!

Python Shell

20
>>> c
30
>>> trio = a, b, c
>>> trio
(10, 20, 30)
>>> x, y, z = trio
>>> x
10
>>> y
20
>>> z
30
>>> i, j = trio
Traceback (most recent call last):
  File "<pyshell#10>", line 1, in <module>
    i, j = trio
ValueError: too many values to unpack
>>> |
Logistic (lab cancelled)

Indexing in Strings

**Indexing in Lists**

Splitting Strings into Lists

String Methods
Strings are collections of characters
Lists are collections of various data types

animals = [ "pig", "hen", "dog", "cat" ]
animals = [ "pig", "hen", "dog", "cat" ]
animals = [ "pig", "hen", "dog", "cat" ]

animals[0]
animals = [ "pig", "hen", "dog", "cat" ]

animals[0]
animals = [ "pig", "hen", "dog", "cat" ]

animals[0]

animals[3]
animals = [ "pig", "hen", "dog", "cat" ]

animals[0]

animals[3]
animals = [ "pig", "hen", "dog", "cat" ]

animals[0]

animals[3]

animals[-3]
animals = [ "pig", "hen", "dog", "cat" ]

animals[0]
animals[3]
animals[-3]
Playing with Python Semantic

farm = ["pig", "dog", "horse", "hen"]

Find as many different ways to print all the animals in the farm, as you can…
Slicing a String

name

0 1 2 3 4 5 6

A L I B A B A
Slicing a String

\[
\text{name} = \text{ALIBABA}
\]

section = name[1:4]
Slicing a String

name

section = name[1:4]

section
name

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>L</td>
<td>I</td>
<td>B</td>
<td>A</td>
<td>B</td>
<td>A</td>
</tr>
</tbody>
</table>

name[0:1]    —>
name    | A L I B A B A A

name[0:1]  ➔  A
name

| A | L | I | B | A | B | A |

name[0:1] → A

name[5:6] →
name

\[
\begin{array}{ccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 \\
A & L & I & B & A & B & A
\end{array}
\]

name[0:1] \quad \rightarrow \quad A

name[5:6] \quad \rightarrow \quad B
name = "ALIBABA"

name[0:1] -> A

name[5:6] -> B

name[-2:-1] -> A
<table>
<thead>
<tr>
<th>name</th>
<th>A</th>
<th>L</th>
<th>I</th>
<th>B</th>
<th>A</th>
<th>B</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>-7</td>
<td>-6</td>
<td>-5</td>
<td>-4</td>
<td>-3</td>
<td>-2</td>
<td>-1</td>
<td></td>
</tr>
</tbody>
</table>

- `name[0:1]`  →  A
- `name[5:6]`  →  B
- `name[-2:-1]`  →  B
- `name[0:-1]`  →  A
name

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>L</td>
<td>I</td>
<td>B</td>
<td>A</td>
<td>B</td>
<td>A</td>
</tr>
</tbody>
</table>

-7 -6 -5 -4 -3 -2 -1

- name[0:1]  -->  A
- name[5:6]  -->  B
- name[-2:-1]  -->  B
- name[0:-1]  -->  ALIBABA
Two Special Slices

\[
\text{part} = \text{name}[3:5]
\]

\[
\text{part} = \text{name}[3:]
\]
Two Special Slices

part = name[ : 5 ] ← from beginning to 5

part = name[ 3 : ] ← from 3 to end, including last
name

0 1 2 3 4 5 6

A L I B A B A

name[ :4]  -->
name[ :4]  →  ALIB
name    \[
\begin{array}{cccccc}
A & L & I & B & A & A \\
\end{array}
\]

name[ :4]  \[\rightarrow\]  \[
\begin{array}{cccc}
A & L & I & B \\
\end{array}
\]

name[ 3: ]  \[\rightarrow\]
```plaintext
D. Thiebaut, Computer Science, Smith College

name

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>L</td>
<td>I</td>
<td>B</td>
<td>A</td>
<td>B</td>
<td>A</td>
</tr>
</tbody>
</table>

name[ :4] —> A L I B

name[ 3: ] —> B A B A B A
```
Interesting Property

name = "Some string of characters"

name2 = name[ : 7] + name[ 7 : ]
Interesting Property

name = "Some string of characters"

name2 = name[ : n] + name[ n : ]

name2 contains the same string as name
Extract the **drive** and **extension** information from a file name:

\[
\text{name} = \text{“H:/Documents/solutionsHw4.doc”}
\]

Replace the “doc” extension by “txt” in the file name:

\[
\text{name} = \text{“H:/Documents/solutionsHw4.doc”}
\]

Get the first and last name of a person and create a computer account with the first letter of the first name, and the last name, concatenated.
Exercises

Transforming dates:
Transform a string, such as “02162015” into 16 Feb 2015.

Get the first and last name from a person, and display a “triangle” made of her full name. For example,
fname = “Maria”
lname = “LUCE”
Output of program:
M
Ma
Mar
Mari
Maria
MariaL
MariaLU
MariaLUCE
We stopped here last time...
Review
• Strings are lists of characters
• Strings are **lists** of characters

name  
\[
\begin{array}{ccccccc}
A & L & I & B & A & B & A \\
\end{array}
\]
• Strings are **lists** of characters

```plaintext
ame  A L I B A B A
```

• Lists are *lists* of items, too!
• Strings are \textbf{lists} of characters

\begin{itemize}
\item name \begin{tabular}{c|c|c|c|c|c|c|c}
A & L & I & B & A & B & A & A
\end{tabular}
\end{itemize}

• Lists are \textit{lists} of items, too!

\begin{verbatim}
farm = [ "dog", "cat", "pig" ]
\end{verbatim}
• Strings are **lists** of characters

  name  \begin{tabular}{c}
  | A | L | I | B | A | B | A |
  \end{tabular}

• Lists are *lists* of items, too!

  farm = [ "dog", "cat", "pig" ]

• They can be indexed, and sliced
• Strings are lists of characters
  
  name  A L I B A B A

• Lists are lists of items, too!

  farm = [ "dog", "cat", "pig" ]

• They can be indexed, and sliced

  name[-1]    name[0:2]

  farm[-1]    farm[0:2]
• Strings are **lists** of characters

```
name A L I B A B A
```

• Lists are **lists** of items, too!

```
farm = [ "dog", "cat", "pig" ]
```

• They can be indexed, and sliced

```
"pig" farm[-1]    farm[0:2]    [ "dog","cat" ]
```

```python
name[-1]    name[0:2]
farm[-1]    farm[0:2]
```
Lists and Strings behave similarly but are different in an important way.
```python
>>> farm = [ "dog", "cat", "pig" ]

>>> farm[ -1 ]
'pig'

>>> farm[ 2:3 ]
['pig']

>>> farm[ 1:3 ]
['cat', 'pig']

>>> farm[ 1 ] = "hen"

>>> farm
['dog', 'hen', 'pig']
```
>>> farm = [ "dog", "cat", "pig" ]

>>> farm[ -1 ]
'pig'

>>> farm[ 2:3 ]
['pig']

>>> farm[ 1:3 ]
['cat', 'pig']

>>> farm[ 1 ] = "hen"

>>> farm
['dog', 'hen', 'pig']

>>> name = "Alibaba"

>>> name[ -1 ]
'a'

>>> name[ 0 ]
'A'

>>> name[ -3:-1 ]
'ab'

>>> name[ 3 ] = 'Z'
Traceback (most recent call last):
  File "<pyshell#22>", line 1, in <module>
    name[3] = 'Z'
TypeError: 'str' object does not support item assignment
We cannot Modify a String!
Strings are Immutable
Logistic (lab cancelled)

Indexing in Strings

Indexing in Lists

String Objects and Methods

Splitting Strings into Lists
Objects
Objects

data
Objects

data

action
Examples
Examples

{"0:1}@{1:1}
Examples

format action
("max", "gmail.com")
Examples

format action
("max", "gmail.com")

max@gmail.com
Examples

"{0:1}@{1:1}".format( "max", "gmail.com" )

format action
("max", "gmail.com")

max@gmail.com
Examples

```
"{0:1}@{1:1}".format( "max", "gmail.com" )
```

format action
("max", "gmail.com")

max@gmail.com
Logistic (lab cancelled)

Indexing in Strings

Indexing in Lists

String Objects and Methods

Splitting Strings into Lists
- `upper()`

"hello there"
• `upper()`
• upper()
- `upper()`

  ```python
  "hello there".upper()
  ```

  "HELLO THERE"
- upper()
- lower()
- center(n)
- capitalize()
- title()
• upper()
• lower()
• center( n )
• capitalize()
• title()

• format( ... , ... )
• find( ... )
• replace( ... , ... )
https://docs.python.org/3.1/library/stdtypes.html?highlight=capitalize#string-methods
Python 3.1.1 (r311:74543, Aug 24 2009, 18:44:04)
[ GCC 4.0.1 (Apple Inc. build 5493)] on darwin
Type "copyright", "credits" or "license()" for more information.

>>> a = ""

demo time

""

>>> |

Ln: 17 Col: 4
Multiple Transformations
"hello there" . upper()

"HELLO THERE"
"hello there".upper()
"hello there".upper()
"hello there".upper().center(20)
Write a program that **prompts** the user for her first and last name, and **prints** both, in the **proper case**, **centered** in 60 spaces.

Write a program that takes a string, where a **phone number** is located. The phone number always **start at Index 7**, and contains 10 numbers (no spaces). **Print the phone number only**, in the form: 

(***xxx***) ***xxx-xxxx***
We stopped here last time...
split()
```python
>>> line = "Snowy day at Smith College"
>>> line
'Snowy day at Smith College'
>>> line.split(' ')
['Snowy', 'day', 'at', 'Smith', 'College']
>>> words = line.split(' ')
>>> words
['Snowy', 'day', 'at', 'Smith', 'College']
>>> for word in words:
    print(word)

Snowy
day
at
Smith
College
>>> for word in line.split(' '):
    print(word)

Snowy
day
at
Smith
College
```
Write a program that takes a **collection of lines** of text, where all the words are separated by **commas**, and prints only the **2nd word** of each line.

Same problem, but this time the program is given a **huge long string** with lines separated by **\n** characters.
Next Week: Functions...