Lecture 12

Lists (& Sequences)
Announcements for Today

(Optional) Reading

• Read 10.0-10.2, 10.4-10.6
• Read all of Chapter 8 for Thu

Prelim, 10/17 at 7:30 pm
• Material up to TODAY
• Study guide is posted
• Times/rooms by last name

Conflict with Prelim time?
• Submit conflict to CMS
• Applies to SDS students too

Assignments

• A2 is now graded
  ▪ Access it in Gradescope
  ▪ Graded out of 50 points
  ▪ Mean: 43.9, Median: 47
  ▪ A: 46 (58%), B: 37 (29%)

• A3 due this Friday
  ▪ Thurs last day for help
  ▪ Will grade over break
Sequences: Lists of Values

### String

- `s = 'abc d'`
  
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td></td>
</tr>
</tbody>
</table>

  - Put characters in quotes
    - Use `'` for quote character
  - Access characters with `[ ]`
    - `s[0]` is 'a'
    - `s[5]` causes an error
    - `s[0:2]` is 'ab' (excludes c)
    - `s[2:]` is 'c d'

### List

- `x = [5, 6, 5, 9, 15, 23]`

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>6</td>
<td>5</td>
<td>9</td>
<td>15</td>
<td>23</td>
</tr>
</tbody>
</table>

  - Put values inside `[ ]`
    - Separate by commas
  - Access **values** with `[ ]`
    - `x[0]` is 5
    - `x[6]` causes an error
    - `x[0:2]` is [5, 6] (excludes 2\textsuperscript{nd} 5)
    - `x[3:]` is [9, 15, 23]
# Sequences: Lists of Values

## String

- `s = 'abc d'`
  
<p>| | | | | |</p>
<table>
<thead>
<tr>
<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>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td></td>
</tr>
</tbody>
</table>

- Put characters in quotes
  - Use `\'` for quote character

- Access characters with `[]`
  - `s[0]` is 'a'
  - `s[5]` causes an error
  - `s[0:2]` is 'ab' (excludes c)
  - `s[2:]` is 'c d'

## List

- `x = [5, 6, 5, 9, 15, 23]`
  
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<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>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>5</td>
<td>9</td>
<td>15</td>
<td>23</td>
</tr>
</tbody>
</table>

- Put values inside `[ ]`

- Access values with `[]`
  - `x[0]` is 5
  - `x[6]` causes an error
  - `x[0:2]` is [5, 6] (excludes 2\textsuperscript{nd} 5)
  - `x[3:]` is [9, 15, 23]

---

Sequence is name given to both
Lists Have Methods Similar to String

\[ x = [5, 6, 5, 9, 15, 23] \]

- **index(value)**
  - Return position of the value
  - **ERROR** if value is not there
  - \( x.index(9) \) evaluates to 3

- **count(value)**
  - Returns number of times value appears in list
  - \( x.count(5) \) evaluates to 2

But you get length of a list with a regular function, not method:

\[ \text{len}(x) \]
Representing Lists

Wrong

x = [5, 6, 7, -2]

Box is “too small” to hold the list

Correct

x = id1

Put list in a “folder”

id1

Variable holds id

Unique tab identifier

x = [5, 7, 4, -2]
Lists vs. Class Objects

List

- Attributes are indexed
  - Example: x[2]

<table>
<thead>
<tr>
<th>x</th>
<th>id2</th>
</tr>
</thead>
<tbody>
<tr>
<td>id2</td>
<td>list</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>-2</td>
</tr>
</tbody>
</table>

RGB

- Attributes are named
  - Example: c.red

<table>
<thead>
<tr>
<th>c</th>
<th>id3</th>
</tr>
</thead>
<tbody>
<tr>
<td>id3</td>
<td>RGB</td>
</tr>
<tr>
<td>red</td>
<td>128</td>
</tr>
<tr>
<td>green</td>
<td>64</td>
</tr>
<tr>
<td>blue</td>
<td>255</td>
</tr>
</tbody>
</table>
When Do We Need to Draw a Folder?

- When the value **contains** other values
  - This is essentially want we mean by ‘object’
- When the value is **mutable**

<table>
<thead>
<tr>
<th>Type</th>
<th>Container?</th>
<th>Mutable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>float</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>str</td>
<td>Yes*</td>
<td>No</td>
</tr>
<tr>
<td>Point3</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RGB</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>list</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Lists are Mutable

- **List assignment:**
  \[
  \text{<var>}[\text{<index>}] = \text{<value>}
  \]
  - Reassign at index
  - Affects folder contents
  - Variable is unchanged

- Strings cannot do this
  - \text{s = 'Hello World!'}
  - \text{s[0] = 'J'}  \text{ERROR}
  - String are immutable

- \text{x = [5, 7, 4, -2]}
  - \text{x[1] = 8}
Lists are Mutable

- **List assignment:**
  
  \(<\text{var}>[<\text{index}>] = <\text{value}>\)
  
  - Reassign at index
  - Affects folder contents
  - Variable is unchanged

- **Strings cannot do this**
  
  - \(s = 'Hello World!'\)
  - \(s[0] = 'J'\) **ERROR**
  - String are **immutable**

- \(x = [5, 7, 4, -2]\)
  
  - \(x[1] = 8\)

- \(x = [5, 7, 4, -2]\)
  
  - \(x[1] = 8\)
Slice Assignment

• Can *embed* a new list inside of a list
  ▪ **Syntax:** `<var>[<start>:<end>] = <list>`
  ▪ Replaces that range with content of list

• **Example:**

  ```python
  >>> a = [1,2,3]
  >>> b = [4,5]
  >>> a[:2] = b
  >>> a
  [4, 5, 3]
  ```
**Lists Share Methods with Strings**

\[ x = [5, 6, 5, 9, 15, 23] \]

- **index(value)**
  - Return position of the value
  - **ERROR** if value is not there
  - `x.index(9)` evaluates to 3

- **count(value)**
  - Returns number of times value appears in list
  - `x.count(5)` evaluates to 2

These are **immutable** methods
List Methods Can Alter the List

- **append(value)**
  - A *procedure method*, not a fruitful method
  - Adds a new value to the end of list
  - `x.append(-1)` *changes* the list to `[5, 6, 5, 9, -1]`

- **insert(index, value)**
  - Put the value into list at index; shift rest of list right
  - `x.insert(2,-1)` *changes* the list to `[5, 6, -1, 5, 9,]`

- **sort()**
List Methods Can Alter the List

\[ x = [5, 6, 5, 9] \]

- **append(value)**
  - A **procedure method**, not a fruitful method
  - Adds a new value to the end of list
  - `x.append(-1)` changes the list to \([5, 6, 5, 9, -1]\)

- **insert(index, value)**
  - Put the value into list at index; shift rest of list right
  - `x.insert(2, -1)` changes the list to \([5, 6, -1, 5, 9,]\)

- **sort()**

What do you think this does?
5.1. More on Lists

The list data type has some more methods. Here are all of the methods of list objects:

`list.append(x)`
Add an item to the end of the list. Equivalent to `a[len(a):] = [x]`.

`list.extend(iterable)`
Extend the list by appending all the items from the iterable.

`list.insert(i, x)`
Insert an item at a given position. The first argument is the index of the element before which to insert, so `a.insert(0, x)` inserts at the front of the list, and `a.insert(len(a), x)` is equivalent to `a.append(x)`.

`list.remove(x)`
Remove the first item from the list whose value is equal to `x`. It raises a `ValueError` if there is no such item.

`list.pop([i])`
Remove the item at the given position in the list, and return it. If no index is specified, `a.pop()` removes and returns the last item in the list. (The square brackets around the `i` in the method signature denote that the parameter is optional, not that you should type square brackets at that position. You will see this notation frequently in the Python Library Reference.)
Lists and Functions: Swap

1. `def swap(b, h, k):
2.     """ Swaps b[h] and b[k] in b
3.     Precond: b is a mutable list, h, k are valid positions"
4.     temp= b[h]
5.     b[h]= b[k]
6.     b[k]= temp

Swaps b[h] and b[k], because parameter b contains name of list.

swap(x, 3, 4)
Lists and Functions: Swap

1. `def swap(b, h, k):`
2. """ Swaps b[h] and b[k] in b
3. **Precond**: b is a mutable list,
4. h, k are valid positions"
5. temp= b[h]
6. b[h]= b[k]
7. b[k]= temp

Swaps b[h] and b[k], because parameter b contains name of list.

```
swap(x, 3, 4)
```

```
<table>
<thead>
<tr>
<th>swap</th>
<th>h</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>id4</td>
<td>3</td>
</tr>
<tr>
<td>temp</td>
<td>6</td>
<td>k</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>id4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>
```

```
x  id4
```

10/2/18
Lists & Sequences
Lists and Functions: Swap

1. \textbf{def} \texttt{swap}(b, h, k):

2. \\
3. \\
4. \\
5. \\
6. \\
7. \\
8. \\
9. \\
10. \\

\texttt{swap}(x, 3, 4)
Lists and Functions: Swap

1. def swap(b, h, k):
2.     """ Swaps b[h] and b[k] in b
3.     Precond: b is a mutable list,
4.     h, k are valid positions"
5.     temp= b[h]
6.     b[h]= b[k]
7.     b[k]= temp

Swaps b[h] and b[k], because parameter b contains name of list.

swap(x, 3, 4)
List Slices Make Copies

\[ x = [5, 6, 5, 9] \]
\[ y = x[1:3] \]

\[ \begin{array}{c|c|c}
0 & 5 & \text{id5} \\
1 & 6 & \text{list} \\
2 & 5 & \text{id5} \\
3 & 9 & \text{list} \\
\end{array} \]

\[ \begin{array}{c|c|c}
0 & 6 & \text{id6} \\
1 & 5 & \text{list} \\
\end{array} \]

\[ \text{copy = new folder} \]
Exercise Time

• Execute the following:
  >>> x = [5, 6, 5, 9, 10]
  >>> x[3] = -1
  >>> x.insert(1,2)

• What is x[4]?

A: 10
B: 9
C: -1
D: ERROR
E: I don’t know
Exercise Time

- Execute the following:
  
  ```
  >>> x = [5, 6, 5, 9, 10]
  >>> x[3] = -1
  >>> x.insert(1,2)
  ```

- What is x[4]?

- Execute the following:
  
  ```
  >>> x = [5, 6, 5, 9, 10]
  >>> y = x[1:]
  >>> y[0] = 7
  ```

- What is x[1]?

- What is x[4]?

  - A: 7
  - B: 5
  - C: 6
  - D: ERROR
  - E: I don’t know
Exercise Time

- Execute the following:
  
  ```python
  >>> x = [5, 6, 5, 9, 10]
  >>> x[3] = -1
  >>> x.insert(1,2)
  ```

- What is `x[4]`?

- Execute the following:
  
  ```python
  >>> x = [5, 6, 5, 9, 10]
  >>> y = x[1:]
  >>> y[0] = 7
  ```

- What is `x[1]`?

- What is `x[4]`?
Lists and Expressions

- List brackets [] can contain expressions
- This is a list **expression**
  - Python must evaluate it
  - Evaluates each expression
  - Puts the value in the list
- Example:
  ```
  >>> a = [1+2,3+4,5+6]
  >>> a
  [3, 7, 11]
  ```

- Execute the following:
  ```
  >>> a = 5
  >>> b = 7
  >>> x = [a, b, a+b]
  ```
- What is x[2]?  
  
  **A:** 'a+b'  
  **B:** 12  
  **C:** 57  
  **D:** ERROR  
  **E:** I don’t know
Lists and Expressions

- List brackets [] can contain expressions
- This is a list **expression**
  - Python must evaluate it
  - Evaluates each expression
  - Puts the value in the list
- Example:
  ```python
  >>> a = [1+2, 3+4, 5+6]
  >>> a
  [3, 7, 11]
  ```

- Execute the following:
  ```python
  >>> a = 5
  >>> b = 7
  >>> x = [a, b, a+b]
  ```

- What is x[2]?
  ```
  12
  ```
Lists of Objects

- List positions are variables
  - Can store base types
  - But cannot store folders
  - Can store folder identifiers
- Folders linking to folders
  - Top folder for the list
  - Other folders for contents
- Example:
  >>> r = introcs.RGB(255,0,0)
  >>> g = introcs.RGB(0,255,0)
  >>> b = introcs.RGB(0,0,255)
  >>> x = [r,g,b]
Lists of Objects

- List positions are variables
  - Can store base types
  - But cannot store folders
  - Can store folder identifiers
- Folders linking to folders
  - Top folder for the list
  - Other folders for contents
- Example:
  >>> r = introcs.RGB(255,0,0)
  >>> g = introcs.RGB(0,255,0)
  >>> b = introcs.RGB(0,0,255)
  >>> x = [r,g,b]