Lists and Sequences

[Lecture 10]

[Andersen, Gries, Lee, Marschner, Van Loan, White]
Lecture 10 Announcements

• Prelim 1
  ▪ **Date:** Tuesday, March 14th, 7:30 pm to 9:00 pm
  ▪ Submit conflicts immediately through CMS

• A2: You must scan or take a picture of your work to submit it through CMS
  ▪ Since you have been warned to submit early, do not expect that we will accept work that does not make it onto CMS on time.

• Set CMS notifications to receive all emails!
Sequences: Lists of Values

String

- \( s = 'abc d' \)
- 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

- \( x = [5, 6, 5, 9, 15, 23] \)
- 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

- $s = 'abc d'$
  
<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 $[]$
  - Use 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]

---

Sequence is a name we give to both
Lists Have Methods Similar to String

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

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

- `<list>.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:

len(x)
Things that Work for All Sequences

\[
\begin{align*}
\text{s} &= \text{‘slithy’} \\
x &= [5, 6, 9, 6, 15, 5]
\end{align*}
\]

### Built-in Functions

- `len(s) → 6`
- `s[4] → “h”`
- `s[1:3] → “li”`
- `s[3:] → “thy”`
- `s[–2] → “h”`

### Methods

- `s.index(‘s’) → 0`
- `s.count(‘t’) → 1`
- `x.index(5) → 0`
- `x.count(6) → 2`
- `len(x) → 6`
- `x[4] → 15`
- `x[1:3] → [6, 9]`
- `x[3:] → [6, 15, 5]`
- `x[–2] → 15`
- `x + [1, 2] → [5, 6, 9, 6, 15, 5, 1, 2]`
- `x * 2 → [5, 6, 9, 6, 15, 5, 6, 9, 6, 15, 5]`

### Slicing

- `s[1:3] → “li”`
- `s[3:] → “thy”`
- `s[–2] → “h”`
- `s + ‘ toves’ → “slithy toves”`
- `s + ‘ toves’ → “slithy toves”`
Difference: Lists Can Hold Any Type

- A list of integers: 5, 6, 8, 9, 15, 23
- A list of objects of class Point: id1, id2, id5, id4, id3
- A list of values of various types: 5, ‘a’, ‘joy’, 24.3, id1, id3, 0, id2
Representing Lists

Wrong

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

Correct

\[ x = \text{id1} \]

- Variable holds id
- Unique tab identifier
- Indices:
  - 0: 5
  - 1: 7
  - 2: 4
  - 3: -2
Lists vs. Class Objects

List

- Attributes are indexed
  - Example: x[2]

```
  x
  id2
  
  id2
  list
  0 5
  1 7
  2 4
  3 -2
```

Objects

- Attributes are named
  - Example: p.x

```
P
  id3
  
  id3
  Point3
  x 1.0
  y 2.0
  z 3.0
```
List Assignment

- **Format:**
  - `<var>[<index>] = <value>`
    - Reassign at index
    - Affects folder contents
    - Variable is unchanged

- `x = [5, 7, 4, -2]`
  
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>7</td>
<td>4</td>
<td>-2</td>
</tr>
</tbody>
</table>

- `x[1] = 8`

- `id1`
  
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>7</td>
<td>4</td>
<td>-2</td>
</tr>
</tbody>
</table>
List Assignment

- **Format:**
  
  `<var>[<index>] = <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`

- `s = 'Hello World!'`

- `s[0] = 'J'` **ERROR**

- String are **immutable**
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]?
  >>> 12
List Methods Can Alter the List

x = [5, 6, 5, 9]

• `<list>.append(<value>)`
  - Procedure, 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]

• `<list>.insert(<index>,<value>)`
  - Procedure, not a fruitful method
  - Puts value into list at index; shifts rest of list right
  - x.insert(2,-1) changes the list to [5, 6, -1, 5, 9]

• `<list>.sort()`

What do you think this does?
Clicker Exercise

• 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
import geom
p = geom.Point3(1.0, 2.0, 3.0)
q = geom.Point3(3.0, 4.0, 5.0)
swap_x(p, q)

import geom
p = geom.Point3(1.0, 2.0, 3.0)
q = geom.Point3(3.0, 4.0, 5.0)
swap(p, q)

def swap_x(p, q):
    t = p.x
    p.x = q.x
    q.x = t

def swap(p, q):
    t = p
    p = q
    q = t

swaps p.x and q.x

DOES NOT swap global p and q
def swap(b, h, k):
    """Procedure swaps b[h] and b[k] in b
    Precondition: b is a mutable list, h and k are valid positions in the list"""
    temp = b[h]
    b[h] = b[k]
    b[k] = temp

What gets printed?
A: 5
B: 6
C: Something else
D: I don’t know

x = [5,4,7,6,5]
swap(x, 3, 4)
print x[3]
def swap(b, h, k):
    """Procedure swaps b[h] and b[k] in b
    Precondition: b is a mutable list, h
    and k are valid positions in the list"""
    temp = b[h]
    b[h] = b[k]
    b[k] = temp

x = [5, 4, 7, 6, 5]
swap(x, 3, 4)
print(x[3])
def swap(b, h, k):
    """Procedure swaps b[h] and b[k] in b
    Precondition: b is a mutable list, h and k are valid positions in the list""
    temp = b[h]
    b[h] = b[k]
    b[k] = temp

x = [5, 4, 7, 6, 5]
swap(x, 3, 4)
print x[3]
def swap(b, h, k):
    """Procedure swaps b[h] and b[k] in b
    Precondition: b is a mutable list, h and k are valid positions in the list"""
    temp = b[h]
    b[h] = b[k]
    b[k] = temp

x = [5, 4, 7, 6, 5]
swap(x, 3, 4)
print(x[3])
def swap(b, h, k):
    """Procedure swaps b[h] and b[k] in b
    Precondition: b is a mutable list, h and k are valid positions in the list"""
    temp = b[h]
    b[h] = b[k]
    b[k] = temp

x = [5, 4, 7, 6, 5]
swap(x, 3, 4)
print x[3]
def swap(b, h, k):
    """Procedure swaps b[h] and b[k] in b
    Precondition: b is a mutable list, h and k are valid positions in the list"""
    temp = b[h]
    b[h] = b[k]
    b[k] = temp

x = [5, 4, 7, 6, 5]
swap(x, 3, 4)
print x[3]

What gets printed?
A: 5
B: 6
C: Something else
D: I don’t know
### List Slices Make Copies

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

\[ y = x[1:3] \]

**Diagram:**
- **x** (id5)
  - list
  - 0: 5
  - 1: 6
  - 2: 5
  - 3: 9
- **y** (id6)
  - list
  - 0: 6
  - 1: 5

**Copy:**
- new folder
Clicker Exercises

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

  - What is `x[1]`?

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

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

  - What is `x[1]`?

A: 7  
B: 5  
C: 6  
D: ERROR  
E: I don’t know
Lists of Objects

- List positions are variables
  - Can store base types
  - But cannot store folders
  - Can store folder ids
- Folders linking to folders
  - Top folder for the list
  - Other folders for contents
- Example:
  ```python
  >>> p1 = Point3(1.0, 2.0, 3.0)
  >>> p2 = Point3(4.0, 5.0, 6.0)
  >>> p3 = Point3(7.0, 8.0, 9.0)
  >>> x = [p1,p2,p3]
  ```
Lists of Objects

• Example:
  >>> p1 = Point3(1.0, 2.0, 3.0)
  >>> p2 = Point3(4.0, 5.0, 6.0)
  >>> p3 = Point3(7.0, 8.0, 9.0)
  >>> x = [p1,p2,p3]

• How do I get this y?
  >>> x[1].y
Lists and Strings Go Hand in Hand

text.split(<sep>): return a list of the words in text (separated by <sep>, or whitespace by default)
<sep>.join(words): concatenate the items in the list of strings words, separated by <sep>.

text = 'A sentence is just
na list of words'
words = text.split() ['A', 'sentence', 'is', 'just', 'a', ...]
lines = text.split('\n') ['A-sentence-is-just-a...']
print '-' . join(words) 'A-sentence-is-just-a...
print '-' . join(lines[0].split() + lines[1].split())

returns a list of two strings

'A-sentence-is-just a-list-of-words'
Example: Poetry

- Can we “read” a poem and count the number of:
  - characters
  - words
  - lines
  - stanzas
Iteration

• To process a list, you often want to do the same thing to each item in the list. One way to do this:

  - The map function:
    
    \[
    \text{map}(\langle \text{function} \rangle, \langle \text{list} \rangle)
    \]

    Call the function once for each item in the list, with the list item as the argument, and put the return values into a list.
The Map Function

- **map**(⟨*function*⟩, ⟨*list*⟩)
  - Function has to have exactly 1 parameter
  - Otherwise, get an error
  - Returns a new list
- Does the same thing as

```python
def map(f, x):
    result = []  # empty list
    for y in x:
        result.append(f(y))
    return result
```

\[ \text{map}(f, x) \rightarrow [f(x[0]), f(x[1]), \ldots, f(x[n-1])] \]

maps the function \( f \) once for each item

\[ \text{map(len, ['a', 'bc', 'defg'])} \]

returns \[1, 2, 4\]