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 'cd'

List
- `x = [5, 6, 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 2nd 5)
  - `x[3:]` is [9, 15, 23]

Lists Have Methods Similar to String

- `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: `len(x)`

Representing Lists

Wrong
- `x = 5, 6, 7, -2`
  - Box is "too small" to hold the list

Correct
- `x = id1`
  - Variable holds id
  - Box in a "folder"
  - Unique tab identifier

Lists vs. Class Objects

List
- Attributes are indexed
  - Example: `x[0]`

RGB
- Attributes are named
  - Example: `x.red`

When Do We Need to Draw a Folder?

- When the value **contains** other values
  - This is essentially what 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>list</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>Point</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: `<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[1] = 8`
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, 8, 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, -1]`

- **sort()**
  - What do you think this does?

List Slices Make Copies

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

- Execute the following:
  - `>>> x = [5, 6, 6, 9, 10]`
  - `>>> x[3] = -1`
  - `>>> x.insert(1, 2)`
  - What is `x[4]`?

Exercise Time

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

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+2,5+4,5+6]`
  - `>>> a`
    - `[8, 9, 11]`
  - `>>> 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 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:
  - `>>> x = cornell.RED`
  - `>>> y = cornell.BLUE`
  - `>>> g = cornell.GREEN`
  - `>>> x = (r, h, g)`