Lists: Sequences of Objects

<table>
<thead>
<tr>
<th>String</th>
<th>List</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>s = 'abc def'</code></td>
<td><code>x = [5, 6, 9, 16, 23]</code></td>
</tr>
<tr>
<td>• Put characters in quotes</td>
<td>• Put items inside [ ]</td>
</tr>
<tr>
<td>• Access characters with [ ]</td>
<td>• Access items with [ ]</td>
</tr>
<tr>
<td>• <code>s[0]</code> is <code>'a'</code></td>
<td>• <code>x[0]</code> is 5</td>
</tr>
<tr>
<td>• <code>s[0:2]</code> is <code>'ab'</code> (excludes <code>c</code>)</td>
<td>• <code>x[0:2]</code> is <code>[5, 6]</code> (excludes 9)</td>
</tr>
<tr>
<td>• <code>s[3]</code> is <code>'e'</code></td>
<td>• <code>x[4]</code> is 23</td>
</tr>
</tbody>
</table>

Examples:
- `s[3:]` causes an error
- `s[0:2]` is `[5, 6]` (excludes 9)
- `s[0]` is `'a'`
- `s[4]` is `23`

Attributes are indexed
- `x[0]` is 5
- `x[4]` is 23

Put characters in quotes
- `s = "Hello World!"` causes an error
- `s = 'Hello World!'`

Attributes are named
- `x[1] = 8`
- `x.insert(1, 2)`

Strings vs. Objects With Attributes

<table>
<thead>
<tr>
<th>List</th>
<th>Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Attributes are indexed</td>
<td>• Attributes are named</td>
</tr>
<tr>
<td>• Example: <code>x[0]</code></td>
<td>• Example: <code>p.x</code></td>
</tr>
<tr>
<td><code>id6</code></td>
<td><code>p.id7</code></td>
</tr>
<tr>
<td><code>0 1 2 3 4 5 6 7</code></td>
<td><code>x 3.0 y 4.0 z 5.0</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>x[1] = 8</code></td>
<td><code>x.insert(1, 2)</code></td>
</tr>
</tbody>
</table>

Clicker Exercise

- Execute the following:
  - `x = [5, 6, 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

- Execute the following:
  - `y = x`
  - `y[1] = 7`
  - What is `x[1]`?

  - A: 7
  - B: 5
  - C: 6
  - D: ERROR
  - E: I don’t know
Lists and Functions: Swap

```python
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
```

Slicing Lists Makes Copies

```python
x = [5, 6, 5, Point(3, 4, 5)]
y = x[1:3]
```

Clicker Exercise

• 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, Point(1, 2, 3), 6]
  >>> y = x[1:]
  >>> y[0].x = 7
  ```
  • What is x[1].x?
  A: 7
  B: 5
  C: 6
  D: ERROR
  E: I don’t know

Lists and Strings Go Hand in Hand

```python
text = 'A sentence is just
       a list of words'
lines = text.split('
')
sep = '-'
print sep.join(lines[0].split()) + ' ' + sep.join(lines[1].split())
```

Tuples

```
strings: immutable sequences of characters
        "tuple" generalizes "pair," "triple," "quadruple,..."

lists: mutable sequences of any objects

"tuple" generalizes "pair," "triple," "quadruple,..."

immutable sequences of any objects

• Tuples fall between strings and lists
  * write them with just commas: 42, 4.0, ‘x’
  * often enclosed in parentheses: (42, 4.0, ‘x’)```

Conventionally use lists for:
• long sequences
• homogeneous sequences
• variable length sequences

Conventionally use tuples for:
• short sequences
• heterogeneous sequences
• fixed length sequences