### Lists: Sequences of Objects

#### String
- `s = 'abc de'`
- Access characters with `s[i]`
- Put characters in quotes

#### List
- `x = [5, 6, 9, 6, 15, 5]`
- Access items with `x[i]`
- Put items inside brackets
- Attributes are named
- Attributes are indexed

#### Difference: Lists Hold Any Type

<table>
<thead>
<tr>
<th>List</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="List example" /></td>
<td><img src="image2" alt="String example" /></td>
</tr>
</tbody>
</table>

#### Lists vs. Objects With Attributes

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

- **Point**
  - Attributes are named
  - Example: `p.x`

### Things that Work for All Sequences

- **Difference: Lists are mutable**
  - Their contents can be altered
  - by assignment to list items
  - using methods
  - draw lists as folders
  - can omit type to save space

- **String**
  - `s = 'slithy'`
- **List**
  - `x = [5, 6, 9, 6, 15, 5]`

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

- **What is `x[4]`?**
  - A: 10
  - B: 9
  - C: -1
  - D: ERROR
  - E: I don’t know

### Clicker Exercise

- **Execute the following:**
  - `x = [5, 6, 9, 6, 10]`
  - `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
  >>> y = [5, 6, 5, 9, 10]
  >>> y[0] = 7
  ```
  What is y[0]?
  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[1:]
  >>> y[0] = 7
  ```
  What is x[1]?
  A: 1
  B: 5
  C: 7
  D: ERROR
  E: I don’t know

Lists and Strings: They go together like…

```python
text = 'Rama lama lama\nke ding a de ding a dong'
words = text.split()
lines = text.split('
')
sep = '-'
print(sep.join(words))
print(sep.join(lines[0].split()) + ' ' + sep.join(lines[1].split()))
```

```
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.
```

Tuples

- 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

```
"tuple" generalizes "pair," "triple," "quadruple," …
"list" generalizes "sequence," "vector," "array," …
```

... a horse and carriage? Bread and butter?