Sequences: Lists of Values

String
- \( s = 'abc d' \)
- Put characters in quotes
  * Use \( \backslash \) for quote character
- Access characters with \([\]
  * \([0]\) is 'a'
  * \([5]\) causes an error
  * \([0:2]\) is 'ab' (excludes c)
  * \([2:]\) is 'cd'

List
- \( x = [5, 6, 5, 9, 15, 23] \)
- Put values inside \([\]
  * Separate by commas
- Access values with \([\]
  * \([0]\) is 5
  * \([6]\) causes an error
  * \([0:2]\) is [5, 6] (excludes 2nd 5)
  * \([3:]\) is [9, 15, 23]

Lists Have Methods Similar to 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

Modifying List Contents

- List assignment:
  * Reassign at index
  * Affects folder contents
  * Variable is unchanged
- Strings cannot do this
  * \( s = 'Hello World!' \)
  * \( s[0] = 'J' \) ERROR
  * String are immutable

Exercise: List Assignment

- Assignment copies id into y
  >>> \( x = [5, 7, 4, -2] \)
  >>> y = x
- Execute the assignments:
  >>> \( x[2] = 8 \)
  >>> \( y[2] = 3 \)
- What is value of \( x[2] \)?
  A: 8
  B: 3
  C: id1
  D: I don’t know

List Methods Can Alter the List

- \( x = [5, 6, 9] \)
  - 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 \([5, 6, 9, 1] \)
  - insert(index, value)
    * Put the value into list at index; shift rest of list right
    * \( x.insert(2, 1) \) changes the list to \([5, 6, 1, 9] \)
  - sort()
    * What do you think this does?
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
swap(x, 3, 4)
```

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

List Slices Make Copies

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

How Multidimensional Lists are Stored

- `b = [[9, 6, 4], [6, 7, 7]]`

- `b` holds name of a one-dimensional list
  - Has `len(b)` elements
  - Its elements are (the names of) 1D lists
- `b[1]` holds the name of a one-dimensional list (of ints)
  - Has `len(b[1])` elements

Two Dimensional Lists

<table>
<thead>
<tr>
<th>Table of Data</th>
<th>Images</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2</td>
<td>5 4 7 1</td>
</tr>
<tr>
<td>1 0 4 9 7</td>
<td>2 5 1 2</td>
</tr>
<tr>
<td>3 1 4 2 9</td>
<td>4 1 2 9</td>
</tr>
<tr>
<td>6 7 8 0</td>
<td>6 7 8 0</td>
</tr>
</tbody>
</table>

Slices and Multidimensional Lists

- Only “top-level” list is copied.
- Contents of the list are not altered
- `b = [[9, 6, 4, 5], [7, 7]]`
- `x = b[2]`