Announcements

- **Only if** you cannot write Prelim 1 in person on Mar 30 at 6:30pm Ithaca time or have SDS exam accommodations, do the CMS “assignment” called “Prelim 1 alternate format/time request” (both Parts A & B). Request deadline is Mar 16 11:59pm. Legitimate reasons needed to request online format and/or alternative time
  - Conflicting exam listed on University Evening Prelim Schedule
  - You are not in Ithaca
- “Go to” lab weekly!! Stay on track. Great student:staff ratio!
- A2 due Mar 19 at 11:59pm
- Window to submit A1 revisions closes Mar 20 at 11:59pm

Sequences: Lists of Values

### String
- `s = 'abc def'`
- 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`
- `len(s)` → 5, length of string

### 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\nd 5)
- `x[3:]` is `[9, 15, 23]`
- `len(x)` → 6, length of list

**Sequence** is a name we give to both

Lists Have Methods Similar to String

- `<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 to get the length of a list you use a function, not a class method:

```
len(x)
```

Lists vs. Class Objects

<table>
<thead>
<tr>
<th>List</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes are indexed</td>
<td>Attributes are named</td>
</tr>
<tr>
<td>Example: x[0]</td>
<td>Example: p.x</td>
</tr>
</tbody>
</table>

Representing Lists

**Wrong:**

<table>
<thead>
<tr>
<th>Global Space</th>
<th>Heap Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = [5, 4, -2]</td>
<td>x</td>
</tr>
</tbody>
</table>

**Correct:**

<table>
<thead>
<tr>
<th>Global Space</th>
<th>Heap Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = [5, 7, 4, -2]</td>
<td>x</td>
</tr>
</tbody>
</table>

Indices

- `0`
- `1`
- `2`
- `3`
Lists Can Hold Any Type

Expression evaluates to value; value goes in list

<table>
<thead>
<tr>
<th>id1</th>
<th>id2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Global Space
- list_of_integers
- list_of_strings

Heap Space
- list_of_integers
- list_of_strings

No Really, Lists Can Hold Any Type!

<table>
<thead>
<tr>
<th>id5</th>
<th>id6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Global Space
- list_of_points
- list_of_various_types

Heap Space
- list_of_points
- list_of_various_types

Lists of Objects
- List elements are variables
  - Can store base types and ids
  - Cannot store folders

Global Space
- p1
- p2
- p3
- x

Heap Space
- id1
- id2
- id3
- id4

How do I get this y?

List Methods Can Alter the List

x = [5, 6, 5, 9]
y = [15, 16, 15, 19]

- <list>.append(value)
  - 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)
  - Puts value into list at index; shifts rest of list right
  - y.insert(2, -1) changes the list to [15, 16, -1, 15, 19]

- <list>.sort()

What do you think this does?

List is mutable; strings are not

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

Global Space
- x
- y

Heap Space
- id1
- id2
- id3
- id4

Strings cannot do this
- Strings are immutable

Q1: Insert into list

- 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
Recall: identifier assignment → no swap

```python
import shapes
def swap(p, q):
    tmp = p
    p = q
    q = tmp
p = shapes.Point3(1,2,3)
q = shapes.Point3(3,4,5)
swap(p, q)
```

At the end of `swap`: parameters `p` and `q` are swapped

Global Space

<table>
<thead>
<tr>
<th>Global Space</th>
<th>Heap Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>p id4</td>
<td>id6</td>
</tr>
<tr>
<td>q</td>
<td>id7</td>
</tr>
<tr>
<td>swap</td>
<td></td>
</tr>
<tr>
<td>tmp</td>
<td></td>
</tr>
<tr>
<td>p</td>
<td>id7</td>
</tr>
<tr>
<td>q</td>
<td>id6</td>
</tr>
</tbody>
</table>

Recall: Attribute Assignment → swap!

```python
import shapes
def swap_x(p, q):
    tmp = p.x
    p.x = q.x
    q.x = tmp
p = shapes.Point3(1,2,3)
q = shapes.Point3(3,4,5)
swap_x(p, q)
```

At the end of `swap`: parameters `p` and `q` are unchanged, attributes `x` are swapped

Q2: Swap List Values?

```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
x = [5,4,7,6,8]
swap(x, 3, 4)
```

What gets printed?

A: 8  
B: 6  
C: Something else  
D: I don't know

Q3: List Slicing

• Execute the following:
  ```
  >>> 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

Q4

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

A: 7  
B: 5  
C: 6  
D: ERROR  
E: I don’t know
**Things that Work for All Sequences**

`s = 'slithy'
x = [5, 6, 9, 6, 15, 5]`

- `s.index('s') → 0`
- `s.count('t') → 1`
- `len(s) → 6`
- `s[4] → 't'`
- `s[1:3] → 'th'`
- `s[3:] → 'thy'`
- `s[-2] → 'ty'`
- `s * 'loves' → "slithyloves"`
- `s * 2 → "slithy"`
- `'t' in s → True`

- `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[1:2] → [15]`
- `x[-2] → 15`
- `x * 2 → [5, 6, 9, 6, 15, 5, 6, 9, 6, 15, 5]`
- `'t' in x → True`

---

**Tuples (see lesson video)**

<table>
<thead>
<tr>
<th>strings: immutable sequences of characters</th>
<th>tuples*: immutable sequences of any objects</th>
<th>lists: mutable sequences of any objects</th>
</tr>
</thead>
</table>

- *“tuple” generalizes “pair,” “triple,” “quadruple,” …*

- Tuples fall between strings and lists
  - write them with just commas: `42, 4.0, 'x'`
  - often enclosed in parentheses: `(42, 4.0, 'x')`

**Use tuples for:**
- short sequences
- heterogeneous sequences
- fixed length sequences

---

**Lists and Strings Go Hand in Hand**

- `text.split(<sep>): return a list of words in text (separated by <sep>, or whitespace by default)`
- `words + text.split(): Turns string into a list of words`
- `words: return a list of words`
  - `[A, 'sentence', 'is', 'just', 'a', 'list', 'of', 'words']`
- `lines + text.split(<sep>): Turns string into a list of lines`
- `lines: return a list of lines`
  - `['A sentence is just', 'a list of words']`
- `hyphenated + '-' + join(words): Combines elements with hyphens`
- `hyphenated + '-' + join(lines[0].split()+lines[1].split()): Merges 2 lists, combines elements with hyphens`
- `A-sentence-is-just-a-list-of-words: return a list of words, separated by <sep>`

**Turns string into a list of words**

**Turns string into a list of lines**

**Combines elements with hyphens**

**Merges 2 lists, combines elements with hyphens**

---

**Use lists for:**
- long sequences
- homogeneous sequences
- variable length sequences

---

**Turns string into a list of words**