Announcements

**Prelim 1 conflicts** must be registered in CMS by **noon Friday**. See email from Maria Witlox.

**Assignment 1 revisions:** make sure CMS notifications are set to send you email when your grade is updated.

**Piazza** traffic is getting higher—but if you decide to opt for less email, be sure to check regularly (at least every 12 hours), and always check the night of a deadline. Otherwise you could miss information that you would be sorry to be without!
# Lists: Sequences of Objects

## String
- $s = 'abc\ de'$
  - **Literal:** characters in quotes
    - Use `\` for quote character
  - **Access characters with [ ]**
    - $s[0]$ is 'a'
    - $s[6]$ causes an error
    - $s[0:2]$ is 'ab' (excludes c)
    - $s[2:]$ is 'c\ de'

## List
- $x = [5, 6, 8, 9, 15, 23]$
  - **Literal:** items inside [ ]
    - Separate by commas
  - **Access items with [ ]**
    - $x[0]$ is 5
    - $x[6]$ causes an error
    - $x[0:2]$ is [5, 6] (excludes the 8)
    - $x[3:]$ is [9, 15, 23]

Both are called “sequences”
**Things that Work for All Sequences**

- **s = ‘slithy’**
  - `s.index('s')` → 0
  - `s.count('t')` → 1
  - `len(s)` → 6
  - `s[4]` → “h”
  - `s[1:3]` → “li”
  - `s[3:]` → “thy”
  - `s[-2]` → “h”
  - `s + ' toves'` → “slithy toves”
  - `s * 2` → “slithyslithy”
  - ‘t’ in s → True

- **x = [5, 6, 9, 6, 15, 5]**
  - `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[-2]` → 15
  - `x + [1, 2]` → [5, 6, 9, 6, 15, 5, 1, 2]
  - `x * 2` → [5, 6, 9, 6, 15, 5, 5, 6, 9, 6, 15, 5]
  - 15 in x → True

The smallest $i$ for which $x[i] == 5$

The number of $i$s for which $x[i] == 6$
### Difference: Lists Hold Any Type

- **a list of integers**
  - 0: 5
  - 1: 6
  - 2: 8
  - 3: 9
  - 4: 15
  - 5: 23

- **a list of strings**
  - 0: ‘H’
  - 1: ‘e’
  - 2: ‘l’
  - 3: ‘l’
  - 4: ‘o’
  - 5: ‘ ’
  - 6: ‘World’

- **a list of objects of class Point**
  - 0: id1
  - 1: id2
  - 2: id5
  - 3: id4
  - 4: id3

- **a heterogeneous list**
  - 0: 5
  - 1: ‘a’
  - 2: ‘joy’
  - 3: 24.3
  - 4: id1
  - 5: id3
  - 6: 0
  - 7: id2

**Diagram:**
- `id1`  `id2`  `id3`  `id4`  `id5`
- Point  Point  Point  Point  Point
Difference: Lists are mutable

- Their contents **can be altered**
  - by assignment to list items
    \[
    x = [5, 7, 3, 1]
    x[1] = 8
    \]
  - using methods
    \[
    x.append(2)
    x.extend([3, 4])
    x.insert(5, 6)
    x.sort()
    \]
- Draw lists as folders
  - because they are mutable objects
  - can omit type to save space

Does not work for strings
\[
\text{s = 'Hello World!'}
\]
\[
\text{s[0] = 'J'} \quad \text{ERROR}
\]
\[
\text{s.append('?')} \quad \text{ERROR}
\]

See Python Standard Library for more methods.
Lists vs. Objects With Attributes

List

- Attributes are indexed
  - Example: a[2]

```
a
id6
```

```
id6
0 1 2 3 4 5 6 7
1 2 3 3 4 5 6 8
```

Point

- Attributes are named
  - Example: p.x

```
p
id7
```

```
Point
id7
x 3.0
y 4.0
z 5.0
```
Clicker Exercise

• 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

• Execute the following:
  >>> x = [5, 6, 5, 9, 10]
  >>> y = x
  >>> y[1] = 7

• What is x[1]?
  A: 7
  B: 5
  C: 6
  D: ERROR
  E: I don’t know
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

# Example usage
swap(x, 3, 4)

Swaps b[h] and b[k], because parameter b contains name of list.
Slicing Lists Makes Copies

\[ x = [5, 6, 5, \text{Point}(3, 4, 5)] \]

\[ y = x[1:3] \]

\[ z = x[:] \]
Clicker Exercise

- 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

- Execute the following:
  >>> x = [5, Point(1, 2, 3), 6]
  >>> y = x[1:]
  >>> y[0].x = 7
- What is x[1].x?
  A: 1
  B: 5
  C: 7
  D: ERROR
  E: I don’t know
Lists and Strings Go Hand in Hand

text.split(sep): return a list of the words in text (separated by sep, or whitespace by default)

text = 'A sentence is just \na list of words'
words = text.split()
lines = text.split('\\n')
sep = '-'
print sep.join(words) # ['A', 'sentence', 'is', 'just', 'a', '…']

returns a list of two strings

text = 'A sentence is just
na list of words'
words = text.split() # ['A', 'sentence', 'is', 'just', 'a', '…']
lines = text.split('\\n') # ['A sentence is just a list of words', '…']
sep = '-'
print sep.join(words) # ['A', 'sentence', 'is', 'just', 'a', '…']
print sep.join(lines[0].split()) + ' ' + sep.join(lines[1].split()) # 'A-sentence-is-just-a-list-of-words'

sep.join(words): concatenate the items in the list of strings words, separated by sep.
To process a list, you often want to do the same thing to each item in the list. Two ways to do this:

- The `map` function:
  \[
  \text{map}(\langle \text{function} \rangle, \langle \text{list} \rangle)
  \]
  Call the function once for each item in the list, with the list item as the argument, and put the return values into a list.

- The `for` statement:
  \[
  \text{for} \langle \text{variable} \rangle \text{ in} \langle \text{list} \rangle:
  \langle \text{statements} \rangle
  \]
  Execute the statements once for each item in the list, with the value of the variable set to the list item.
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

Strings:
- immutable sequences of characters

Lists:
- mutable sequences of any objects

Tuples:
- immutable sequences of any objects