The Sequence Abstraction

There isn’t just one sequence class or data abstraction (in Python or in general).
The sequence abstraction is a collection of behaviors:

- **Length.** A sequence has a finite length.
- **Element selection.** A sequence has an element corresponding to any non-negative integer index less than its length, starting at 0.

There is built-in syntax associated with this behavior, or we can use functions. A list is a kind of built-in sequence.

```
for element in s:
    if element == value:
        total = total + 1
```

---

**Sequences**

**Length.** A sequence has a finite length.

**Element selection.** A sequence has an element corresponding to any non-negative integer index less than its length, starting at 0.

```
[2, 7] * digits + 2
[2, 7, 1, 6, 2, 8, 1, 8, 2, 8]
```

```
pairs = [[10, 20], [30, 40]]
pairs[1][0]
```

---

**Lists**

Lists are Sequences

```
>>> digits = [1, 8, 2, 8]
4
>>> len(digits)
4
```

```
>>> [2, 7] * digits + 2
[2, 7, 1, 6, 2, 8, 1, 8, 2, 8]
```

```
>>> pairs = [[10, 20], [30, 40]]
>>> pairs[1]
[30, 40]
```

---

**For Statements**

```
def count(s, value):
    total = 0
    for element in s:
        if element == value:
            total = total + 1
    return total
```

---

**Sequence iteration**

---

**CS 61A Lecture 10**

---

**Announcements**
For Statement Execution Procedure

```python
for <name> in <expression>:
    <suite>
```

1. Evaluate the header `<expression>`, which must yield an iterable value (a sequence)
2. For each element in that sequence, in order:
   A. Bind `<name>` to that element in the current frame
   B. Execute the `<suite>`

Sequence Unpacking in For Statements

```python
>>> pairs = [[1, 2], [2, 2], [3, 2], [4, 4]]
>>> same_count = 0

for x, y in pairs:
    if x == y:
        same_count = same_count + 1

>>> same_count
2
```

A sequence of fixed-length sequences
A name for each element in a fixed-length sequence
Each name is bound to a value, as in multiple assignment

Ranges

```python
>>> range(-2, 2)
[-2, -1, 0, 1]

>>> list(range(4))
[0, 1, 2, 3]
```

A range is a sequence of consecutive integers.
Ranges can actually represent more general integer sequences.

List Comprehensions

```python
>>> letters = ['a', 'b', 'c', 'd', 'e', 'f', 'm', 'n', 'o', 'p']

>>> [letters[i] for i in [3, 4, 6, 8]]
['d', 'e', 'm', 'o']
```

A combined expression that evaluates to a list using this evaluation procedure:
1. Add a new frame with the current frame as its parent
2. Create an empty result list that is the value of the expression
3. For each element in the iterable value of `<iter exp>`:
   A. Bind `<name>` to that element in the new frame from step 1
   B. If `<filter exp>` evaluates to a true value, then add the value of `<map exp>` to the result list

Strings are an Abstraction

```python
'curry = lambda f: lambda x: lambda y: f(x, y)'
```

Representing data:

```python
'200' '1.2e-5' 'False' '[1, 2]'"And, as imagination bodies forth
The forms of things to unknown, and the poet's pen
Turns them to shapes, and gives to airy nothing
A local habitation and a name."
```

Representing programs:

```python
'Demo')
String Literals Have Three Forms

```python
>>> 'I am string!
'I am string!
'>>> "I've got an apostrophe
"I've got an apostrophe
">>> '您好
"您好
">>> "The Zen of Python
claims, Readability counts.
Read more: import this.
"The Zen of Python
claims, Readability counts.
Read more: import this.
```

Strings are Sequences

Length and element selection are similar to all sequences

```python
>>> city = 'Berkeley'
>>> len(city)
8
>>> city[3]
'k'
```

However, the "in" and "not in" operators match substrings

```python
>>> 'here' in "Where's Waldo?"
True
>>> 234 in [1, 2, 3, 4, 5]
False
>>> [2, 3, 4] in [1, 2, 3, 4, 5]
False
```

When working with strings, we usually care about whole words more than letters

Dictionaries

Dictionaries are unordered collections of key-value pairs

Dictionary keys do have two restrictions:

- A key of a dictionary cannot be a list or a dictionary (or any mutable type)
- Two keys cannot be equal; there can be at most one value for a given key

This first restriction is tied to Python’s underlying implementation of dictionaries

The second restriction is part of the dictionary abstraction

If you want to associate multiple values with a key, store them all in a sequence value

Limitations on Dictionaries

Dictionaries are unordered collections of key-value pairs

Dictionary keys do have two restrictions:

- A key of a dictionary cannot be a list or a dictionary (or any mutable type)
- Two keys cannot be equal; there can be at most one value for a given key

This first restriction is tied to Python’s underlying implementation of dictionaries

The second restriction is part of the dictionary abstraction

If you want to associate multiple values with a key, store them all in a sequence value

Length and element selection are similar to all sequences

```python
>>> city = 'Berkeley'
>>> len(city)
8
>>> city[3]
'k'
```

However, the "in" and "not in" operators match substrings

```python
>>> 'here' in "Where's Waldo?"
True
>>> 234 in [1, 2, 3, 4, 5]
False
>>> [2, 3, 4] in [1, 2, 3, 4, 5]
False
```

When working with strings, we usually care about whole words more than letters

Dictionaries

Dictionaries are unordered collections of key-value pairs

Dictionary keys do have two restrictions:

- A key of a dictionary cannot be a list or a dictionary (or any mutable type)
- Two keys cannot be equal; there can be at most one value for a given key

This first restriction is tied to Python’s underlying implementation of dictionaries

The second restriction is part of the dictionary abstraction

If you want to associate multiple values with a key, store them all in a sequence value

Limitations on Dictionaries

Dictionaries are unordered collections of key-value pairs

Dictionary keys do have two restrictions:

- A key of a dictionary cannot be a list or a dictionary (or any mutable type)
- Two keys cannot be equal; there can be at most one value for a given key

This first restriction is tied to Python’s underlying implementation of dictionaries

The second restriction is part of the dictionary abstraction

If you want to associate multiple values with a key, store them all in a sequence value