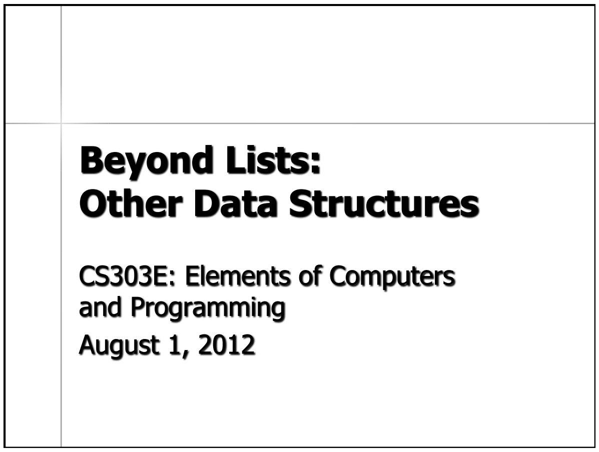
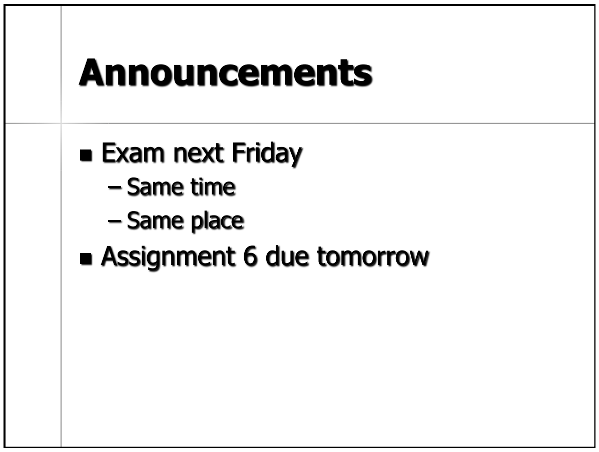


Beyond Lists: Other Data Structures

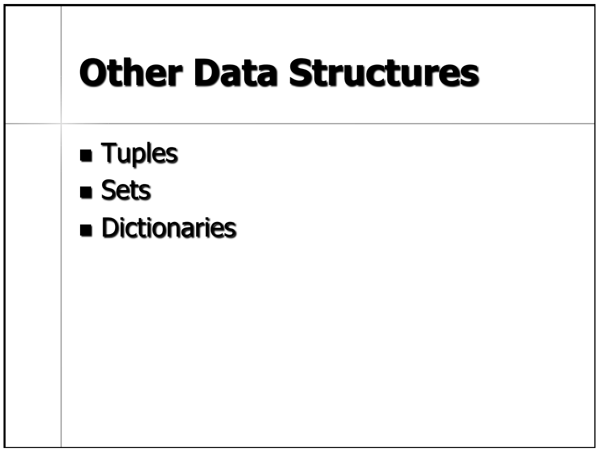
CS303E: Elements of Computers and Programming

August 1, 2012



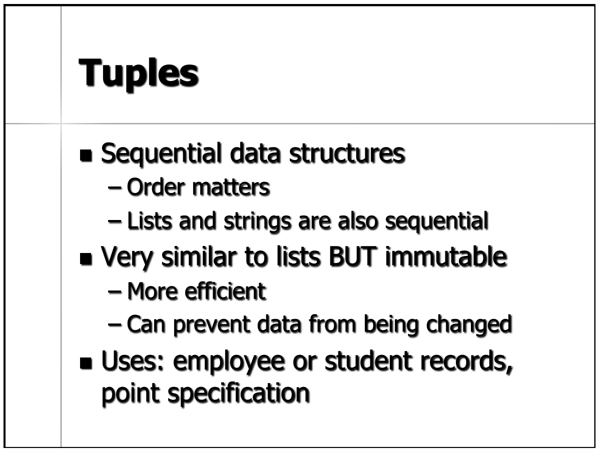
Announcements

- Exploring next Friday
  - Same time
  - Same place
- Assignment 6 due tomorrow



Other Data Structures

- Tuples
- Sets
- Dictionaries



Tuples

- Sequential data structures
  - Order matters
  - Lists and strings are also sequential
- Very similar to lists BUT immutable
  - More efficient
  - Can prevent data from being changed
- Uses: employee or student records, point specification

Tuple Representation

Like lists, but with parentheses instead of square brackets

```
myTuple = (1,2,3)
myTuple = (“a”, “b”, 3)
myTuple = (1,2,1,2,3)
```

Tuple Operations

Operation	Description
`[]`	`Indexing`
`[:]`	`Slicing`
`+`	`Concatenation`
`*`	`Repetition`
`len()`	`Length`
`for elem in myTuple` OR `for i in range(len(myTuple))` then access with `[]`	`Traverse`
`elem in myTuple`	`Membership`

Packing and Unpacking Tuples

- You can also create a tuple by *packing* values together:

```
myTuple = “a”,4,"hello"
```

- And you can unpack them:

```
letter, num, word = myTuple
```

Converting Tuples

- Convert from tuple to list using `list`

```
list(myTuple)
```

- Convert from list to tuple using `tuple`

```
tuple(myList)
```

Sets

A set is an unordered collection with no duplicates

- Similar to mathematical sets

Set Representation

- Create from a list:

```
mySet = set([1,2,3,4])

myList = [1,2,3,4,5]
mySet = set(myList)
```

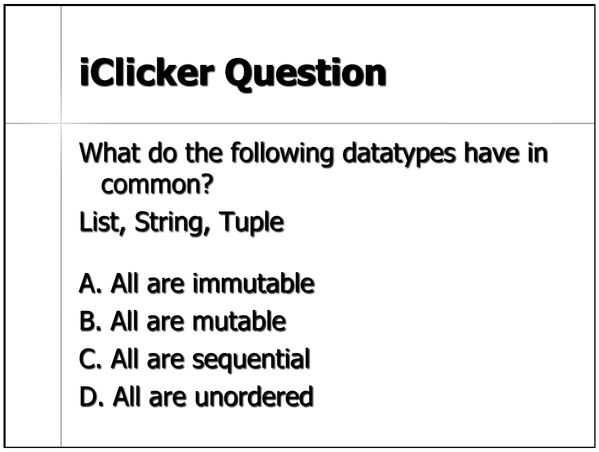
- Duplicate elements will be eliminated and the ordering lost

Set Operations

- Union (|)
  - Elements in one or both sets
- Intersection (&)
  - Elements in both sets
- Difference (–)
- Symmetrical Difference (^)
  - Elements in one but not both sets

More Set Operations

Operation	Description
`<setName>.add(<elem>)`	Add an element to the set
`<setName>.clear()`	Remove all elements from the set
`len()`	Length
`for elem in mySet`	Traverse
`elem in mySet`	Membership



iClicker Question

What do the following datatypes have in common?

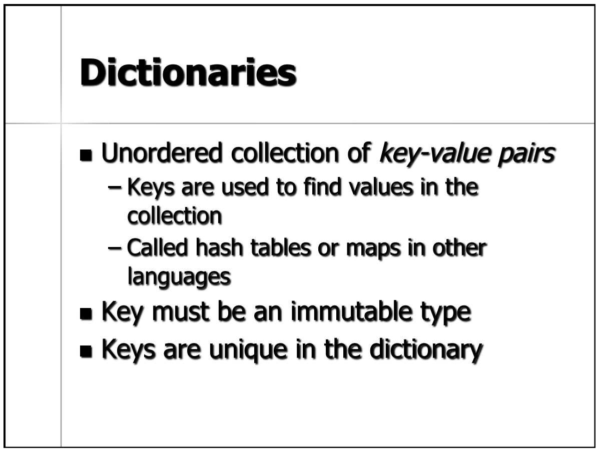
List, String, Tuple

A. All are immutable

B. All are mutable

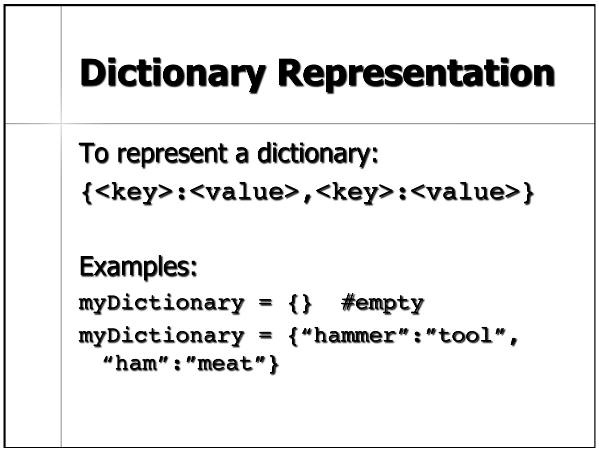
C. All are sequential

D. All are unordered



Dictionaries

- Unordered collection of *key-value pairs*
  - Keys are used to find values in the collection
  - Called hash tables or maps in other languages
- Key must be an immutable type
- Keys are unique in the dictionary



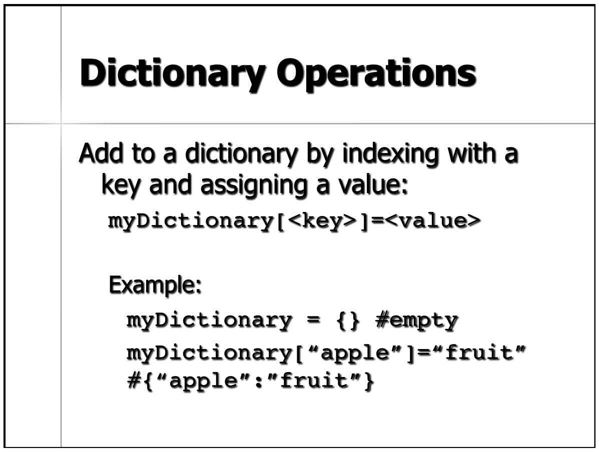
Dictionary Representation

To represent a dictionary:

```
{<key>:<value>,<key>:<value>}
```

Examples:

```
myDictionary = {}  #empty
myDictionary = {“hammer”:”tool“,
  “ham”:”meat”}
```



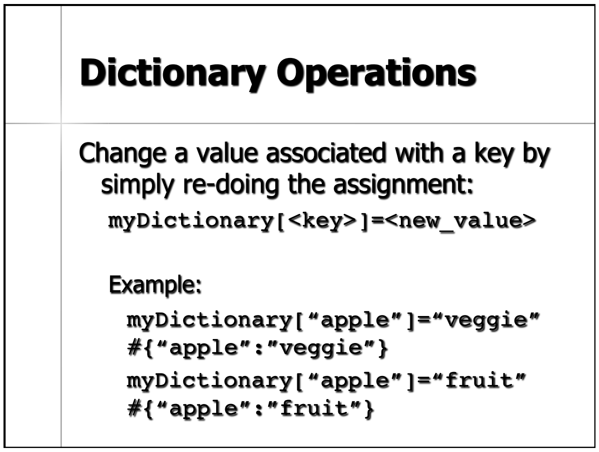
Dictionary Operations

Add to a dictionary by indexing with a key and assigning a value:

```
myDictionary[<key>]=<value>
```

Example:

```
myDictionary = {} #empty
myDictionary[“apple”]=“fruit”
#{“apple”:”fruit”}
```



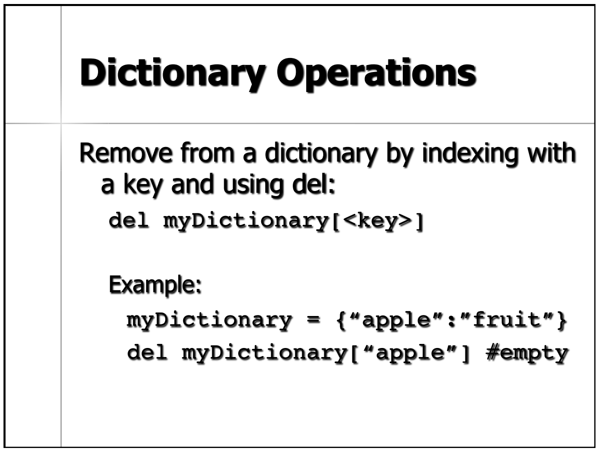
Dictionary Operations

Change a value associated with a key by simply re-doing the assignment:

```
myDictionary[<key>]=<new_value>
```

Example:

```
myDictionary[“apple”]=“veggie”
#{“apple”:”veggie”}
myDictionary[“apple”]=“fruit”
#{“apple”:”fruit”}
```



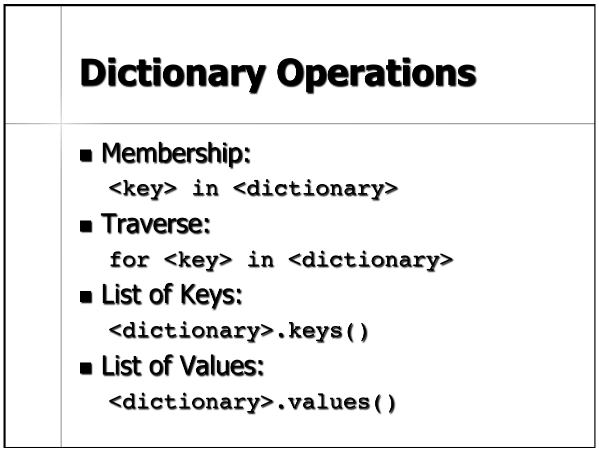
Dictionary Operations

Remove from a dictionary by indexing with a key and using del:

```
del myDictionary[<key>]
```

Example:

```
myDictionary = {“apple”:”fruit”}
del myDictionary[“apple”] #empty
```



Dictionary Operations

- Membership:

```
<key> in <dictionary>
```

- Traverse:

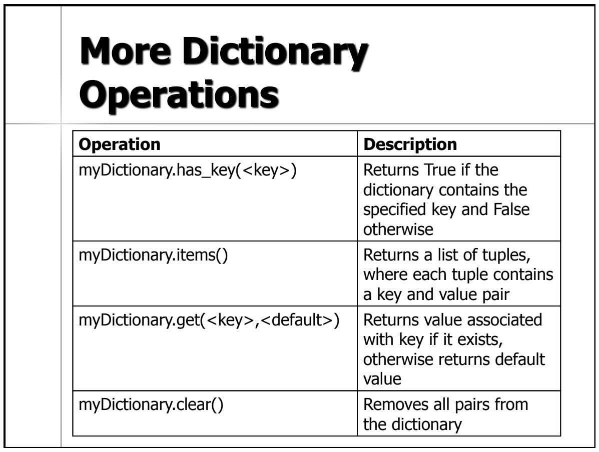
```
for <key> in <dictionary>
```

- List of Keys:

```
<dictionary>.keys()
```

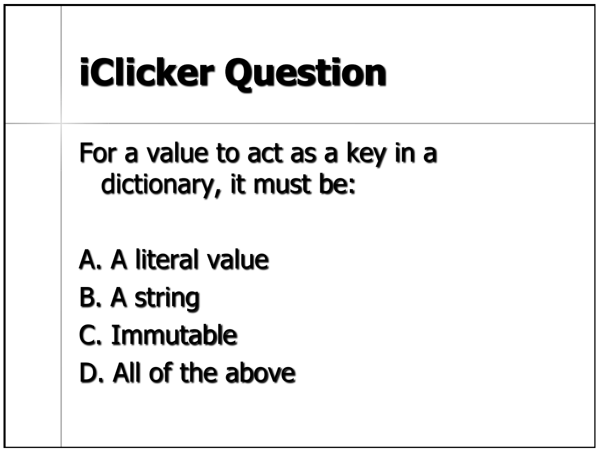
- List of Values:

```
<dictionary>.values()
```



More Dictionary Operations

Operation	Description
myDictionary.has_key(<key>)	Returns True if the dictionary contains the specified key and False otherwise
myDictionary.items()	Returns a list of tuples, where each tuple contains a key and value pair
myDictionary.get(<key>,<default>)	Returns value associated with key if it exists, otherwise returns default value
myDictionary.clear()	Removes all pairs from the dictionary



iClicker Question

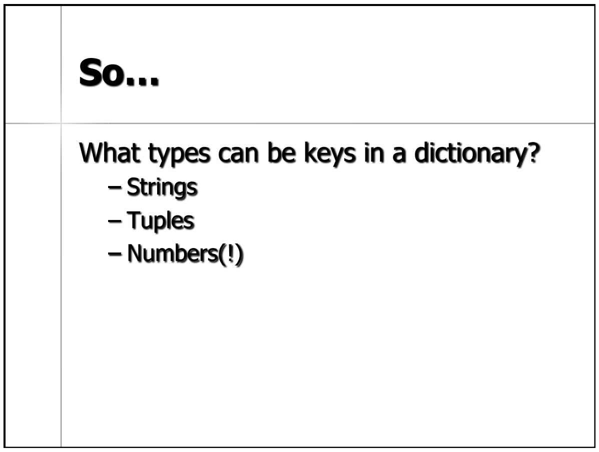
For a value to act as a key in a dictionary, it must be:

A. A literal value

B. A string

C. Immutable

D. All of the above



So…

What types can be keys in a dictionary?

- Strings
- Tuples
- Numbers(!)