Dictionaries or mappings

• A dictionary maps each key to a value
• Order does not matter
• Given a key, can look up a value
  – Given a value, cannot look up its key
• No duplicate keys
  – Multiple keys may map to the same value
• Keys and values are Python values
  – Keys must be immutable (not a list, set, or dict)
• Can add key → value mappings to a dictionary
  – Can also remove (less common)
Dictionary syntax in Python

d = { }  
d = dict()  
us_wars_by_end = {  
    1783: "Revolutionary",  
    1848: "Mexican",  
    1865: "Civil"  }
us_wars_by_name = {  
    "Civil" : [1861, 1865],  
    "Mexican" : [1846, 1848],  
    "Revolutionary" : [1775, 1783]  }

• Syntax just like arrays, for accessing and setting:
  
  us_wars_by_end[1783]    ⇒ "Revolutionary"  
  us_wars_by_end[1783][1:10]    ⇒ “evolution”  
  us_wars_by_name["WWI"] = [1917, 1918]
Creating a dictionary

```python
>>> state = {"Atlanta": "GA", "Seattle": "WA"}

>>> phonebook = dict()
>>> phonebook["Alice"] = "206-555-4455"
>>> phonebook["Bob"] = "212-555-2211"

>>> atomicnumber = {}
>>> atomicnumber["H"] = 1
>>> atomicnumber["Fe"] = 26
>>> atomicnumber["Au"] = 79
```
Accessing a dictionary

```python
>>> atomicnumber = {"H":1, "Fe":26, "Au":79}
>>> atomicnumber["Au"]
79
>>> atomicnumber["B"]
Traceback (most recent call last):
  File "<pyshell#102>", line 1, in <module>
    atomicnumber["B"]
KeyError: 'B'
>>> atomicnumber.has_key("B")
False
>>> atomicnumber.keys()
['H', 'Au', 'Fe']
>>> atomicnumber.values()
[1, 79, 26]
>>> atomicnumber.items()
[('H', 1), ('Au', 79), ('Fe', 26)]
```

Good for iteration (for loops)

```
for key in mymap.keys():
    val = mymap[key]
    ... use key and val
```

```
for key in mymap:
    val = mymap[key]
    ... use key and val
```

```
for (key,val) in mymap.items():
    val = mymap[key]
    ... use key and val
```
Modifying a dictionary

```python
us_wars1 = {
    "Revolutionary" : [1775, 1783],
    "Mexican" : [1846, 1848],
    "Civil" : [1861, 1865] }

us_wars1["WWI"] = [1917, 1918]  # add mapping
del us_wars_by_name["Mexican"]  # remove mapping
```
Dictionary exercises

• Convert a list to a dictionary:
  • Given [5, 6, 7], produce {5:25, 6:36, 7:49}

• Reverse a dictionary:
  – Given {5:25, 6:36, 7:49}, produce {25:5, 36:6, 49:7}

• What does this do?

  squares = { 1:1, 2:4, 3:9, 4:16 }
squares[3 + 3]
squares[2 + 2]
A list is like a dictionary

• A list maps an integer to a value
  – The integers must be a continuous range 0..i

```python
mylist = ['a','b','c']
mylist[1] => 'b'
mylist[3] = 'c'    # error!
```

• In what ways is a list more convenient than a dictionary?
• In what ways is a list less convenient than a dictionary?
Not every value is allowed to be a key

- Keys must be immutable values
  - int, float, bool, string, tuple
  - not: list, set, dictionary
- Goal: only dictionary operations change the keyset
  - after “mydict[x] = y”, mydict[x] ⇒ y
  - if a == b, then mydict[a] == mydict[b] 
    These conditions should hold until mydict is changed
- Mutable keys can violate these goals

```python
list1 = ['a', 'b']
list2 = list1
list3 = ['a', 'b']
mydict = {}
mydict[list1] = "z"  # Hypothetical; actually illegal in Python
mydict[list3] ⇒ "z"
list2.append("c")
mydict[list1] ⇒ ???
mydict[list3] ⇒ ???
```