

Jin-Soo Kim  
(jinsoo.kim@snu.ac.kr)

Systems Software &  
Architecture Lab.  
Seoul National University

Dec 16 – 20, 2019

# Dictionaries



# Sequence vs. Collection

- Both can hold a bunch of values in a single "variable"
- Sequence
  - Sequence has a deterministic ordering
  - Index their entries based on the position
  - Lists, strings, tuples
- Collection
  - No ordering
  - Sets, dictionaries

# Dictionaries

- Unordered collections of arbitrary objects (key-value pairs)
- Accessed by key, not offset
- Variable-length, heterogeneous, and arbitrarily nestable
- Python's most powerful data structure

```
menu = {'spam':9.99, 'egg':0.99}  
a = {1:'a', 1:'b', 2:'a'}  
b = {'food':{'ham':1, 'egg':2}}  
c = {'food':['spam', 'ham', 'egg']}  
emptydict = {}
```

# Lists vs. Dictionaries

- Dictionaries are like lists except that they use **keys** instead of numbers to look up values

```
>>> lst = list()
>>> lst.append(21)
>>> lst.append(1)
>>> print(lst)
[21, 1]
>>> lst[0] = 23
>>> print(lst)
[23, 1]
```

```
>>> dct = dict()
>>> dct['age'] = 21
>>> dct['course'] = 1
>>> print(dct)
{'course': 1, 'age': 21}
>>> dct['age'] = 23
>>> print(dct)
{'course': 1, 'age': 23}
```

# Counters with a Dictionary

- One common use of dictionaries is **counting** how often we see something

```
>>> lastname = dict()
>>> lastname['kim'] = 1
>>> lastname['lee'] = 1
>>> print(lastname)
{'kim': 1, 'lee': 1}
>>> lastname['kim'] = lastname['kim'] + 1
>>> print(lastname)
{'kim': 2, 'lee': 1}
```

# Dictionary Tracebacks

- It is an error to reference a key which is not in the dictionary
- We can use the **in** operator to see if a key is in the dictionary

```
>>> lastname = dict()
>>> lastname['kim'] = 1
>>> print(lastname['park'])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'park'
>>> 'kim' in lastname
True
>>> 'park' in lastname
False
```

# Counting with the in Operator

- When we encounter a new name, we need to add a new entry in the dictionary
- If this is the second or later time we have seen the name, we simply add one to the count in the dictionary under that name

```
counts = dict()
names = ['kim', 'lee', 'park', 'kim', 'park', 'jang']
for name in names:
    if name not in counts:
        counts[name] = 1
    else:
        counts[name] = counts[name] + 1
print(counts)
```

# Counting with get()

- `dict.get(key[, default])`
  - Return the value of *key* if *key* is in the dictionary, else *default*
  - If *default* is not given, it defaults to **None**
  - Never raises a **KeyError**

```
counts = dict()
names = ['kim', 'lee', 'park', 'kim', 'park', 'jang']
for name in names:
    counts[name] = counts.get(name, 0) + 1
print(counts)
```

```
{'kim': 2, 'lee': 1, 'park': 2, 'jang': 1}
```



# Counting Pattern

- Split the line into words
- Loop through the words
- Use a dictionary to track the count of each word independently

```
counts = dict()
line = input('Enter a line: ')

words = line.split()

print('Words:', words)
print('Counting...')
for word in words:
    counts[word] = counts.get(word, 0) + 1
print(counts)
```

# Counting Pattern: Example

```
$ python wordcount.py
```

```
Enter a line: the clown ran after the car and the car ran into  
the tent and the tent fell down on the clown and the car
```

```
Words: ['the', 'clown', 'ran', 'after', 'the', 'car', 'and',  
'the', 'car', 'ran', 'into', 'the', 'tent', 'and', 'the',  
'tent', 'fell', 'down', 'on', 'the', 'clown', 'and', 'the',  
'car']
```

```
Counting...
```

```
{'the': 7, 'clown': 2, 'ran': 2, 'after': 1, 'car': 3, 'and':  
3, 'into': 1, 'tent': 2, 'fell': 1, 'down': 1, 'on': 1}
```

# Loops over Dictionaries

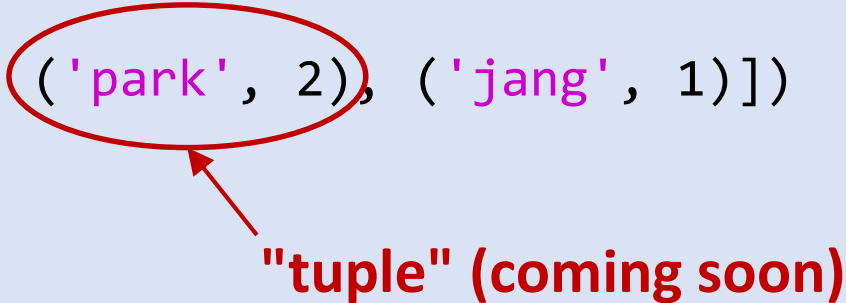
- Even though dictionaries are not stored in order, we can write a **for** loop that goes through all the entries in a dictionary
- Actually it goes through all of the **keys** in the dictionary

```
>>> counts = {'kim': 2, 'lee': 1, 'park': 2, 'jang': 1}
>>> for key in counts:
...     print(key, counts[key])
kim 2
lee 1
park 2
jang 1
```

# Retrieving Lists of Keys and Values

- Use `dict.keys()`, `dict.values()`, and `dict.items()`
- You can loop over them!

```
>>> counts = {'kim': 2, 'lee': 1, 'park': 2, 'jang': 1}
>>> print(counts.keys())
dict_keys(['kim', 'lee', 'park', 'jang'])
>>> print(counts.values())
dict_values([2, 1, 2, 1])
>>> print(counts.items())
dict_items([('kim', 2), ('lee', 1), ('park', 2), ('jang', 1)])
>>> total_count = 0
>>> for count in counts.values():
...     total_count += count
```



"tuple" (coming soon)

# Looping over `dict.items()`

- Loop through the key-value pairs using **two** iteration variables
- The first variable is the **key** and the second is the corresponding **value**

```
>>> counts = {'kim': 2, 'lee': 1, 'park': 2, 'jang': 1}
>>> total_count = 0
>>> for k, v in counts.items():
...     print(k, v)
...     total_count += v
kim 2
lee 1
park 2
jang 1
>>> print(total_count)
6
```

# Counting Words

```
filename = input('Enter file: ')
f = open(filename)

counts = dict()
for line in f:
    words = line.lower().split()
    for word in words:
        counts[word] = counts.get(word, 0) + 1

topcount = None
topword = None
for word, count in counts.items():
    if topcount == None or count > topcount:
        topword = word
        topcount = count

print(topword, topcount)
```

Enter file: genesis.txt  
and 3630