Python Dictionaries

Chapter 9

hat is a Collection?

ection is nice because we can put more than one valu
carry them all around in one convenient package

ave a bunch of values in a single "variable"

o this by having more than one place "in" the variable

ve ways of finding the different places in the variable

What is not a "Collection"

f our variables have one value in them - when we put
the variable - the old value is overwritten

```
S python

darwin
>>> x = 2
>>> x = 4
>>> print x
4
```

Story of Two Collections

ear collection of values that stay in order

nary

ag" of values, each with its own label

ionaries

Dictionaries

aries are Python's most powerful data collection

aries allow us to do fast database-like operations in Python

aries have different names in different languages

ciative Arrays - Perl / PHP

erties or Map or HashMap - Java

erty Bag - C# / .Net

Dictionaries

ndex their entries
on the position in
t

naries are like bags -
der

e index the things we
the dictionary with a
up tag"

```
>>> purse = dict()
>>> purse['money'] = 12
>>> purse['candy'] = 3
>>> purse['tissues'] = 75
>>> print purse
{'money': 12, 'tissues': 75
>>> print purse['candy']
3
>>> purse['candy'] = purse[
>>> print purse
{'money': 12, 'tissues': 75
```

nparing Lists and Dictiona

naries are like lists except that they use keys instead o

ers to look up values

```
lst = list()
lst.append(21)
lst.append(183)
print lst
, 183]
lst[0] = 23
print lst
, 183]
```

```
>>> ddd = dict()
>>> ddd['age'] = 2
>>> ddd['course']
>>> print ddd
{'course': 182, 'a
>>> ddd['age'] = 2
>>> print ddd
{'course': 182, 'a
```

```
lst = list()
lst.append(21)
lst.append(183)
print lst
 183]
lst[0] = 23
print lst
 183]
```

List

Key	Va
[0]	2
[1]	1

```
ddd = dict()
ddd['age'] = 21
ddd['course'] = 182
print ddd
urse': 182, 'age': 21}
ddd['age'] = 23
print ddd
urse': 182, 'age': 23}
```

Dictiona

Key	V
['course']	
['age']	

ctionary Literals (Constan

ary literals use curly braces and have a list of key :valu

make an empty dictionary using empty curly braces

```
jj = { 'DI'     : 1 , 'Team' : 42, 'jan': 100
rint jjj
': 100, 'DI':     1, 'Team': 42}
oo = { }
rint ooo
```

Most Common Name?

quard

n

n

cwen

marquard

zhen

csev

cwen

zhen

csev

marq

zhen

Most Common Name?

quard

n

n

cwen

marquard

zhen

csev

cwen

zhen

csev

marq

zhen

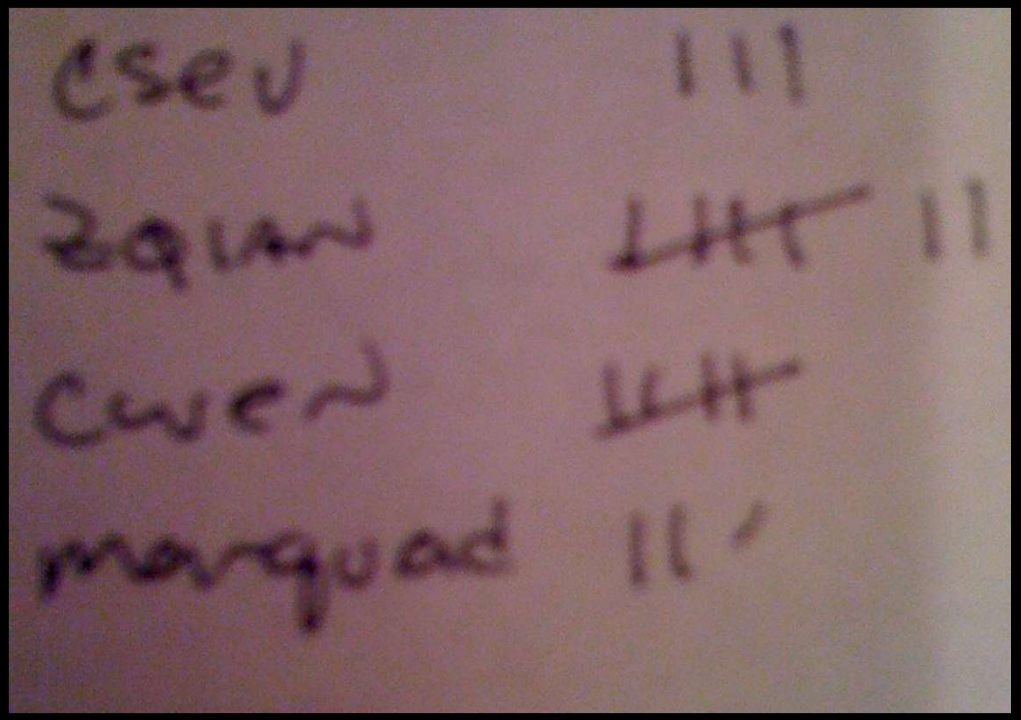
Most Common Name?

quard

n

n

cwen



csev

cwen

zhen

csev

marc

zhen

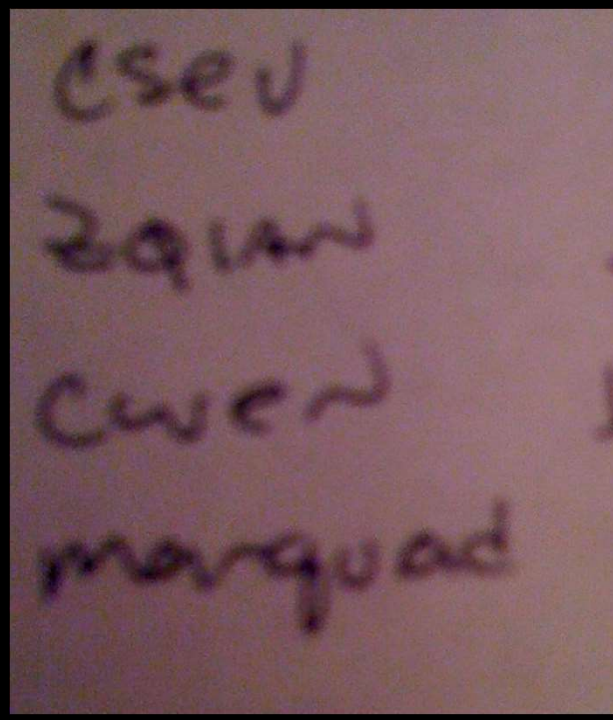
ny Counters with a Diction

ommon use of dictionary is

ing how often we "see" something

```
cc = dict()
cc['csev'] = 1
cc['cwen'] = 1
rint ccc
v': 1, 'cwen': 1}
cc['cwen'] = ccc['cwen'] + 1
rint ccc
v': 1, 'cwen': 2}
```

Key



Dictionary Tracebacks

n error to reference a key which is not in the dictionary

n use the in operator to see if a key is in the dictionar

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

When we see a new name

we encounter a new name, we need to add a new ent
ary and if this the second or later time we have seen t
ply add one to the count in the dictionary under that

```
dict()
'csev', 'cwen', 'csev', 'zqian', 'cwen']
in names :
me not in counts:
unts[name] = 1
:
ounts[name] = counts[name] + 1
nts
```

e get method for dictionar

ttern of checking to see
is already in a
ary and assuming a
value if the key is not
so common, that there
thod called get() that
is for us

```
if name in co
    x = counts
else :
    x = 0
```

```
x = counts.get(
```

lue if key does not exist
nd no Traceback).

{'csev': 2, 'zqian': 1

mplified counting with ge

use get() and provide a default value of zero when th
in the dictionary - and then just add one

```
= dict()
= ['csev', 'cwen', 'csev', 'zqian', 'cwen']
me in names :
unts[name] = counts.get(name, 0) + 1
counts
```

Default

{'csev': 2, 'zqian': 1

mplified counting with ge

```
t()
ev', 'cwen', 'csev', 'zqian', 'cwen']
names :
ame] = counts.get(name, 0) + 1
```

Counting Pattern

```
lict()
ter a line of text:'
_input('')

ne.split()

rds:', words

nting...'
n words:
s[word] = counts.get(word,0) + 1
nts', counts
```

The general pattern to
words in a line of text
the line into words, th
through the words an
dictionary to track the
each word independ

Counting Words

```
unt.py
of text:
 after the car and the car ran into the tent

fell down on the clown and the car
, 'clown', 'ran', 'after', 'the', 'car',
 'car', 'ran', 'into', 'the', 'tent', 'and',
, 'fell', 'down', 'on', 'the', 'clown',
 'car']

: 3, 'on': 1, 'ran': 2, 'car': 3, 'into': 1,
clown': 2, 'down': 1, 'fell': 1, 'the': 7,
```

```
)
 line of text:'
ut('')
plit()

, words
g...'

rds:
d] = counts.get(word,0) + 1
, counts
```

```
python wordcount.py
Enter a line of text:
the clown ran after the car
into the tent and the tent f
the clown and the car

Words: ['the', 'clown', 'ran',
'car', 'and', 'the', 'car', 'ran', 'i
'tent', 'and', 'the', 'tent', 'fell'
'the', 'clown', 'and', 'the', 'ca
Counting...

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

finite Loops and Dictionar

ough dictionaries are not stored in order, we can writ

at goes through all the entries in a dictionary - actuall

all of the keys in the dictionary and looks up the valu

```
unts = { 'chuck' : 1 , 'fred' : 42, 'jan': 1
r key in counts:
  print key, counts[key]

0
1
2
```

eving lists of Keys and Va

get a list of
ues, or
oth) from a
ry

```
>>> jjj = { 'DI'    : 1 , 'Team' : 42,
>>> print list(jjj)
['May', 'DI',    'Team']
>>> print jjj.keys()
['May', 'DI',    'Team']
>>> print jjj.values()
[100, 1, 42]
>>> print jjj.items()
[('May', 100), ('DI',    1), ('Team',
>>>
```

What is a 'tuple'? - com

nus: Two Iteration Variabl

p through the
ue pairs in a
ary using *two*
n variables

eration, the first
e is the key and
ond variable is
esponding value
key

```
>>> jjj = { 'DI'     : 1 , '
'May': 100}
>>> for aaa,bbb in jjj.item
...          print aaa, bbb
...
May 100
DI 1
Team 42
>>>
```

```
nput('Enter file:')
n(name)
e.read()
.split()

t()
words:
rd] = counts.get(word,0) + 1

one
ne
nt in counts.items():
unt is None or count > bigcount:
ord = word
ount = count

d, bigcount
```

```
python word
Enter file: wo
to 16
```

```
python word
Enter file: clo
the 7
```

Summary

- t is a collection?
- versus Dictionaries
- onary constants
- most common word
- g the get() method
- Hashing, and lack o
- Writing dictionary
- Sneak peek: tuples
- Sorting dictionaries