Dictionaries in Python

Zelle, 11.6

Learning Python, 8.3, 8.4

1

Dictionary Basics

- A dictionary can be created by listing key-value pairs inside curly braces
  - Keys and values are joined with ':'
- Main function: Given a key, look up the associated value
- Python stores dictionaries in a way that makes key lookup very efficient (uses a "hash table")
- Do not use dictionaries if you want to keep items in a certain order
- Useful operations: check if a given key is in a dictionary, return keys, return values, delete entries, update values (see Zelle Appendix)

2

Wednesday, April 15, 2009

```
D2 = {"Tom": "497-3467", "Barb": "341-456-
7865", "Bob": ['494-1144', '413-7765']}

D1 = {}
D1["Jim"] = "no phone"
D1["Paul"] = ["463-1234", "447-8754"]

len(D1)

D1["Jim"] = "556-2344"
D1["Mary"] = "no public number"
D1["Ann"] = "447-6755"
```

```
D1.keys()
D1.values()
D1.items()

D1.sort()      # gives error
list1 = D1.items()
list1.sort()

del D1["jim"]   # gives error
del D1["Jim"]

for entry in D1:
    print entry

if "Paul" in D1:
    print D1["Paul"]

D1.get("Mary")
D1.get("Pam", "not in file")
```


Clicker question

birds = {'canada goose' : 183, 'long-tailed jaeger' : 71, 'snow goose' : 63, 'northern fulmar' : 1}

Which expression generates an error?

A. birds['blue jay'] = 9

B. birds['canada goose'] + birds['snow goose']

C. if 'eagle' in birds:
       print 'eagles have been seen'

D. **del birds['eagle']**

Clicker Question

```
>>> D1 = {'Korb':46184, 'Hambrusch':41831, 'Vitek':69644}
>>> D1['Korb'] = [46184, 46003, 413-5667]
>>> D1
{'Vitek': 69644, 'Hambrusch': 41831, 'Korb': [46184, 46003, -5254]}
```

Which does not return 3?

A. len(D1 ['Korb'])
B. len(D1)
C. **3*len(D1['Vitek'])**
D. len(D1.keys())

Dictionaries: main properties

- Searching is easy and efficient (access to an entry is by key)
- Concatenation and slicing don't apply (no ordering exists)
- One can sort the keys and the values
  - can also define ones own comparisons
- Use when lists and arrays don't apply

Birds, with Dictionaries (countbirds.py)

```
bird_count = {}
filename = "bird_observations.txt"

infile = open(filename, 'r')
for line in infile:
    name = string.lower(line.strip())
    if name in bird_count:
        bird_count[name] = bird_count[name] + 1
    else:
        bird_count[name] = 1

# print entries in dictionary, one per line
for b in bird_count:
    print b, "\t", bird_count[b]

# print only the keys
print
print "keys:", bird_count.keys()
```


Creating an inverted dictionary

- Goal: Print the bird observations sorted by frequency
  - from largest to smallest, with all birds having the same frequency listed together
- Frequency entries are integers
  - There are duplicate entries
  - Entries are between 1 and some maximum (not known in advance)

Creating an inverted dictionary

- We could create a list from the bird_count dictionary
  - Entries would look like
    [1, ['a','b','c']], [4, ['d']], …]
- File sortedbycountbirds.py creates an inverted dictionary
  - Could be made more readable by using functions

Word Frequency Problem (Zelle 11.6.3)

- Given a file containing text, count the number of occurrences of each word in the text
- Output a list of words, in sorted order, along with their frequencies
- Dictionaries make this type of problem easy
- Use words as keys and a counter as value:
  {['target': 3], ['the':5], … }
- For each word in the file decide if it is in the dictionary
  - If yes, increment the entry's counter
  - If not, create a new entry

Equivalent statements

"Pythonic"

```
if counts.has_key(w):
        counts[w] = counts[w] + 1
else:
        counts[w] = 1
```

counts[w] = counts.get(w,0) + 1

Returns count[w] if w exists as key in dictionary counts, 0 otherwise

```
if w in counts:
        counts[w] = counts[w] + 1
else:
        counts[w] = 1
```



```
def compareItems((w1,c1), (w2,c2)):
    if c1 > c2:
        return - 1
    elif c1 == c2:
        return cmp(w1, w2)
    else:
        return 1

def main():
    print "This program analyzes word frequency in a file"
    print "and prints a report on the n most frequent words.\n"

    # get the sequence of words from the file
    fname = raw_input("File to analyze: ")
    text = open(fname,'r').read()
    text = string.lower(text)
    for ch in '!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~':
        text = string.replace(text, ch, ' ')
    words = string.split(text)

    counts = {}
    for w in words:
        counts[w] = counts.get(w,0) + 1

    # output analysis of n most frequent words.
    n = input("Output analysis of how many words? ")
    items = counts.items()
    items.sort(compareItems)
    for i in range(n):
        print "%-12s%5d" % items[i]
```

Wordcount Problem - Comments

- Writing ones own "custom" comparison can be very useful
  - Outcomes are -1 (<), 0 (=), 1(>)
  - Give name of function as an argument to the sort method
- Program wordfreq.py asks user for n and outputs the n most frequent words
- What is printed in similar to the inverted dictionary