Compsci 101, Dictionaries

Owen Astrachan
Kristin Stephens-Martinez
March 20, 2018

`P` is for …

- **Python**
  - Whatever you want it to be? Language!!!
- **Parameter**
  - When an argument becomes a variable
- **Power Cycle**
  - Not the last resort. But works
- **P2P**
  - From Torrent to Collaboration

Plan for The Week

- APTs, Assignments, Changes in Due Dates
  - Making sure we're on the same page/calendar
- Dictionary aka map aka hash
  - Very, very fast (key,value) pairs
- Design and Implementation of a game
  - Next two assignments ☺

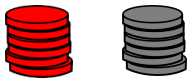
Under the hood

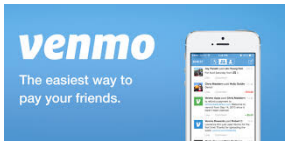
- Using a dictionary is reasonably straight-forward
  - We will be clients, not implementers
  - Efficiency not a large concern in 101
  - We're going along for a ride ☺

Motivation for Dictionary

http://www.cs.duke.edu/csed/pythonapt/networth.html

- If Harry pays Sally $10.23,
  - "Harry:Sally:10.23" and Harry is out $10.23
- How do we extract payer, payee, amount?
  - How to process
  - In Python





Tools We Know About

- Keep track of every person we see
  - Use a list
- Keep track of net worth: money in, money out
  - Use a parallel list
- Maintain invariant: **`names[k] <-> money[k]`**
  - k^{th} name has k^{th} money

INVARIANT

Some APT details

- Given a person's name, if we haven't seen it?
  - Append to **`names`**, append 0 to **`money`**
  - Why must we do both? Invariant, update!
- Find index of person's name in **`names`**
  - Update corresponding entry in **`money`**
- Use $$ * 100 to avoid floating point issues

Seen parallel lists before

- Solution outlined is reasonable, efficiencies?
  - How long does it take to find index of name?
  - Why is the answer "it depends"?
- **`list.index(elt)`** or **`elt in list`** – fast?
  - What does "fast" mean? Relative to what?
- Such a common idiom most languages support fast alternative: dictionary aka map aka hash …

What is a Dictionary?

- A collection of (key,value) pairs: abstract view
  - Look up key, find the value
- Very, very fast: essentially index by key
  - For list **`a[3]`** takes same time as **`a[3000]`**
- Dictionary? **`d["cake"],d["icecreamcone"]`**
  - Finding the value associated with "cake"

Short Code and Long Time

- See module WordFrequencies.py
  - Find # times each word in a list of words occurs
  - How many times is **`words.count(w)`** called?
    - Why is **`set(words)`** used in list comprehension?
  - We have tuple/pair: word and word-frequency

```
37 def slowcount(words):
38     pairs = [(w,words.count(w)) for w in set(words)]
39     return sorted(pairs)
```


WordFrequencies with Dictionary

- We look at a million words to count # "cats"
  - Then a million words to count # "dogs"
  - Could update with parallel lists, still slow
  - Look at each word once: dictionary!
- Key idea: use word as the "key" to find occurrences, update as needed
  - Syntax similar to **`counter[k] += 1`**

Using fastcount

- Update count if we've seen word before
  - Otherwise it's the first time, occurs once

```
28 def fastcount(words):
29     d = {}
30     for w in words:
31         if w in d:
32             d[w] += 1
33         else:
34             d[w] = 1
35     return sorted(d.items())
```

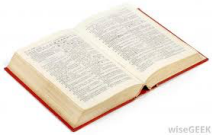
Dictionary Syntax and Semantics

- Creating and using a dictionary

Dictionary Syntax/Function	Meaning
`d = {}`	Initialize empty dictionary d
`d.keys() or d`	Set of keys in dictionary
`d.values()`	Collection of values
`d[key]`	Value associated with key (error to "read" if key not in d)
`d.get(key,dv)`	Value associated wit key (dv if key not in d)
`d.items()`	Collection of (key,value) tuples in d

WOTO

http://bit.ly/101spring18-march20-1



danah boyd

Dr. **danah boyd** is a Principal Researcher at Microsoft Research, … and a Visiting Professor at New York University. Her research is focused on addressing social and cultural inequities by understanding the relationship between technology and society.

I have learned one thing from my research, it's this: social media services like Facebook and Twitter are providing teens with new opportunities to participate in public life, and this, more than anything else, is what concerns many anxious adults."

Solving the Same Problem

VenmoTracker Revisited

- Instead of parallel lists, use a dictionary, e.g., where **`d.items()`** might be

```
[('drew', 10.0),('owen', -30.0),
 ('robert', 10.0),('susan', 10.0)]
```

- How do we extract values to return? What do we know about order in which names appear?
  - From idea to all green

Dictionary Metaphors

- Look up www.cs.duke.edu and get 152.3.140.1
  - Key is URL, value is IP address, browsers
- Look up Duke Blue and get #001A57 or (0,26,87)
  - Key is name of color, value is hex or (r,g,b)
- Look up gmail address, get all your gmail
  - Address + authentication = access (value)

Dictionary Details

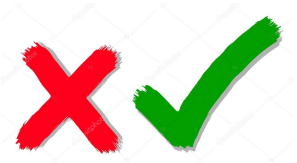
- Initializing or constructing a dictionary
  - `d = {}`
  - `d = {"apple":3, "guava":37}`
  - `d = dict([("owen",62),("kristin",73)])`
  - `d["apple"] = 5`
- Key in dictionary must be immutable so lookup possible
  - String? yes. List? no. Tuple? yes.

DictionaryTimings.py

- **`(key,value)`** pair in **`d.items()`** is tuple, but…
  - We can modify the value given a key, so …
- Function linear: keep [string,counter] ***list***
  - How is variable found used?
- Function binary: if words in order fast lookup
  - Think of a number from 1 to 1000, first guess?
  - What is 2^{10}? Why is this relevant?

List of list pairs in code

- We want to return `[ [count,word],…]` list
  - Makes sorting work by number of words
  - Frequent words at end of list
- Compare to binary, `[[word,count,…]` needed
  - Look up in sorted list, by word!
  - Return `[ [x,y] for [y,x] in data]`

Final WOTO, Correctness

http://bit.ly/101spring18-march20-2