CS2110 Lecture 14 Feb. 24, 2021

- HW 4 due Mar. 4

Last time

- Mutability of lists when passed as arguments to function
- Functions with **side effects**
- Start Ch 12, dictionaries

Today

- More on dictionaries, Ch 12

(last time) Chapter 12: Dictionaries

- Dictionaries are:
  - collections of key – value pairs
- Similar to but importantly different from lists
  - think of lists as *ordered* collection of key-value pairs, where the keys are integers 0, 1, 2, …
  - with dictionaries, the collection is *unordered* but the cool thing is that the *keys can be any immutable values*
  - *E.g.* create dictionary numlegs

    >>> numlegs = { 'frog': 4, 'human': 2, 'ant':6, 'dog':4}
- one important feature of dictionaries is that they provide *very fast* access to values associated with keys despite being more flexible than lists
  - demo dicttest.py

(last time) Dictionaries

- create: { k1:v1, k2:v2, …}
- empty dictionary: {}
- retrieve value: dict[key]
- modify (or insert) value for key: dict[key]=value
- len(d)
- d.keys()
- d.values()
- k in d
- del d[k]
- for key in dict:
- d.get(key, defaultVal) when you don't want possible KeyError for d[key]

Looping over dictionaries with for

If we have a dictionary, d, of names as keys and ages as values, we can compute averages as follows:

```
averageAge = 0
sumOfAges = 0
for nameKey in d:
   sumOfAges += d[nameKey]
if sumOfAges != 0:
   averageAge = sumOfAges / len(d)
print(“The average age is: “, averageAge)
```

A dictionary example

- Text file with info about people – name, birth year, favorite color, weight, home city, home country
- Read and store in dictionary
  - Name as key
  - Subdictionary (and sub-sub-dictionary) for other properties

```
{'birthyear': 1980,
 'favcolor': 'red',
 ...,
 'home': {'city': 'Tokyo', 'country': 'Japan'}
}
```

- Add simple password handling, storing "hash" in dict

Files: ppldata.py, people.text

Related news

- 2015 Turing Award winners: https://www.theguardian.com/science/2016/mar/01/turing-award-whitfield-diffie-martin-hellman-online-commerce
- http://amturing.acm.org/byyear.cfm
- Remember printFirstNPrimes problem? Finding prime factors of big numbers is super Important for crytography. Internet security depends hugely on the fact that there is no known way to find factors of very large numbers quickly

A few little exercises

- Given a list of numbers, find the pair with greatest difference
- Given a list of numbers find the pair with smallest difference
- Given a list of numbers and a target number (call it k), find two numbers (if they exist) in the list that sum to k

Lec14exercises.py has solutions for first two. Has slow (and not completely correct) solution for third one. Can you think of a much faster solution using dictionaries?

Small variants of/questions about third problem

4. Suppose:
   - No dictionaries allowed/available
   - Numbers in lists are know to have limited magnitude. E.g. all numbers between 0 and 10000

   Fast solution?
5. Modify findKPairFast to provide indices/location of found pair in list
6. Question: if we generate a list of, say, 10,000 random numbers between -1,000,000 and 1,000,000 how likely is it that the list contains a pair that sums to k for any k in, say, 0…999?

lec14exercisesB.py

(next time)

Next Time

- DS5 and HW4
- Global variables, tuples (10.26-28), variable length and keyword function parameters
- A bit on list comprehensions (10.22), conditional expresssions, etc.