On your way in...

On the front desk:
1. Graded HW 2s in the folders

Hand-in:
1. Homework 3s due today
   • 2 piles: ID < 50 and ID >= 50
Welcome to CS 134!

Introduction to Computer Science

Iris Howley

-Dictionaries-

Spring 2019
Lists of Lists

• dog2owner = [[‘pixel’,’iris’],[‘wally’,’steve’],[‘tally’,’duane’]]

• What index is the name of Tally’s owner at within dog2owner?
  ▪ Just the owner’s name!

Take a minute to discuss with a partner
Lists of Lists

• dog2owner = 
  [[‘pixel’,‘iris’],[‘wally’,‘steve’],[‘tally’,‘duane’]]

• What index is the name of Tally’s owner at within dog2owner?

1. What is the index of the element of dog2owner that we want?
   ▪ dog2owner[0] → [‘pixel’, ‘iris’]
   ▪ dog2owner[1] → [‘wally’, ‘steve’]

2. What is the index of the element within that element, that we want?
   ▪ [‘tally’, ‘duane’][0] → ‘tally’
   ▪ [‘tally’, ‘duane’][1] → ‘duane’
TODAY’S LESSON

Dictionaries

(a mutable data structure)
l = ['pixel', 'wally', 'tally']

l[1]

  'wally'


d =

  {'pix': 'iris', 'wally': 'steve', 'tally': 'duane'}


d['tally']

  'duane'
Dictionaries

\[d = \{ \text{'pix': 'iris'}, \text{'wally': 'steve'}, \text{'tally': 'duane'} \}\]

- \[d['tally']\] = \text{'duane'}
- \[d['pix']\] = \text{'iris'}
- \[d['wally']\] = \text{'steve'}
Iterating Over Dictionaries

• `d = {'pix': 'iris', 'wally': 'steve', 'tally': 'duane'}`

• for `key` in `d`:
  - `print("{}'s dog is {}".format(d[key], key))`

  - When key is ‘pix’:
    - iris’s dog is pix
      - d[key] is ‘iris’
  - When key is ‘wally’:
    - steve’s dog is wally
      - d[key] is ‘steve’
  - When key is ‘tally’:
    - duane’s dog is tally
      - d[key] is ‘duane’
Dictionary Keys

• \(d = \{1:'hello', 2:2019\}\)
  ▪ Keys can be other types, so can values

• \(d[\text{'good'}] = [\text{'bye'}] * 3\)
  ▪ We can add values mapped to a specified key
  ▪ \(\{1: 'hello', 2: 2019, 'good': ['bye','bye','bye']\}\)
Dictionary Keys

- \( d = \{ \text{`Bill': `Dartmouth'} \} \)
- \( d[\text{`Bill']} = \text{`Stony Brook U'} \)
- \( d \)
  - \( \{ \text{`Bill': `Stony Brook U'} \} \)
  - Only one key with same value! Overwrites!

- \( d[\text{`Bill']} = [\text{`Dartmouth'}, \text{`Stony Brook U'}] \)
- ...But lists can also be dictionary values
Dictionary Keys

- d[('bill l', 'bill j')] = 'williams college'
  - ERROR
- d[('bill l', 'bill j')] = 'williams college'
- d
  - {('bill l', 'bill j'): 'williams college'}

What’s the difference?

Dictionary keys must be immutable types
- int, float, string, bool, tuple, frozenset
Detecting if Something in a Dictionary

- \(d = \{\text{dogs'}: 5, \ 'cats': 1\}\)
- 'cats' in \(d\):
  - True
- 5 in \(d\):
  - False
- 5 in \(d.values()\):
  - True

- \(l = [\text{pix'}, \ 'wally', \ 'tally']\)

- if 'wally' in \(l\):
  - print("Found Wally!")
Dictionaries

- \( \texttt{d} = \texttt{dict}() \)
- \( \texttt{d} = \{4:101, 2:760, 9: 422\} \)
- \( \texttt{list(d)} \)
  - \([4, 2, 9]\)
- \( \texttt{list(d.values())} \)
  - \([101, 760, 422]\)
- \( \texttt{list(d.items())} \)
  - \([(4,101), (2,760), (9,422)]\)

Remember ‘\texttt{pydoc3 dict}’ for more functions!
TODAY’S LESSON

Sets

(a mutable data structure)
Sets

• \( s = \{5, 5, 5, 7, 7, 7, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2\} \)
  • \( s \)
  • \( \{2, 5, 7\} \)

• \( \text{list}(s) \)
  • \([2, 5, 7]\)

• \( \text{tuple}(s) \)
  • \((2, 5, 7)\)

What happened?
No repeats!
Order isn’t preserved!
Sets & Frozensets

- `s = \{4,3,3,3,9,1,1\}`
- `s`
  - `{9, 3, 4, 1}`
- `s[0]`
  - `TypeError: 'set' object does not support indexing`
- `s.add(2019)`
- `s`
  - `{1, 3, 4, 2019, 9}`

- `fs = frozenset(s)`
- `fs`
  - `frozenset({1,3,4,2019,9})`
- `fs[0]`
  - `TypeError: 'frozenset' object does not support indexing`
- `fs.add(2019)`
- `fs`
  - `AttributeError: 'frozenset' object has no attribute 'add'`

'pydoc3 set'
Sets & Dictionaries

• Sets are mutable, so if we want an immutable version:

  • s = \{5, 5, 5, 7, 7, 7, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2\}
  • fs = frozenset(s)

• fs can now be used as a key in a dictionary
Counting Vocabularies

- `f = open(filename)`
- `wordlist = []`
- `for line in f:`
  - `line = line.strip()`
  - `wordlist.extend(line.split())`
- `vocab = set(wordlist)`
- `len(vocab)`

`s = "hello there folks!"

`s.split(" ") ➞ ['hello','there','folks']`
Counting Vocabularies

• Just because you can’t myset[0], doesn’t mean you can’t iterate over elements in a set!

• for item in vocab:
  ▪ if item in ['tom', 'becky']
    ○ print(item)
Algorithms
Fibonacci Sequence

- $\text{fibo}(0) = 0$
- $\text{fibo}(1) = 1$
- $\text{fibo}(n) = \text{fibo}(n-1) + \text{fibo}(n-2)$

- $\text{fibo}(4) = \text{fibo}(3) + \text{fibo}(2)$
  - $= \text{fibo}(2) + \text{fibo}(1) + \text{fibo}(1) + \text{fibo}(0)$
  - $= \text{fibo}(1) + 1 + 1 + 0$
  - $= 1 + 0$
  - $= 3$
Fibonacci Sequence

- \text{fibo}(0) = 1 \text{ call of fibo()}
- \text{fibo}(1) = 1 \text{ call}
- \text{fibo}(2) = 3 \text{ calls}
- \text{fibo}(3) = 5 \text{ calls}
- \text{fibo}(4) = 9 \text{ calls}
- \text{fibo}(5) = 15 \text{ calls}...

- For each increase in \( n \), the number of function calls practically doubles
Speeding Up Fibonacci

(Memoization)

global postit
if n in postit:
    answer = postit[n]
else:
    if n < 2:
        answer = n
    else:
        answer = fibo(n-1) + fibo(n-2)
    postit[n] = answer
return answer
QUESTIONS?
Leftover Slides
Why Lists of Lists?

Images

Games

Mathematics

(1) `pixel-click` event from pixel.vue
i.e. `colors['color-name'] = [255, 255, 255]`

(2) Canvas updates the color dictionary.
(3) colors of pixels are updated.
Hashing
Hashing
Hashing

FIND:
Hashing

FIND:
Hashing

• We could organize all words in memory by the letter they start with...

• But words that start with ‘A’ could be numerous

• Compared to words that start with ‘Z’
  ▪ ...Soft of like arranging clothes by color

• Hashing is a different way of mapping items to make them easier to find
Hashing

• Other concerns
  ▪ Running out of space in the pile you’ve assigned
  ▪ Placing shirts in the wrong pile

• Stored in the order that makes it easiest to look them up