MapReduce, Dictionaries, List Comprehensions

Special thanks to Scott Shawcroft, Ryan Tucker, and Paul Beck for their work on these slides.

Why didn't NaN == NaN?

Last time we found that

float(“nan”) == float(“nan”) is False

NaN is “nothing”, does not equal anything, even itself.

To find out if a value is NaN you have to use isnan

```
>>> from math import *
>>> n = float("nan")
>>> isnan(n)
True
```

MapReduce

Framework for processing huge datasets on certain kinds of distributable problems

Map Step:

- master node takes the input, chops it up into smaller sub-problems, and distributes those to worker nodes.

- worker node may chop its work into yet small pieces and redistribute again

MapReduce

Framework for processing huge datasets on certain kinds of distributable problems

Map Step:

- master node takes the input, chops it up into smaller sub-problems, and distributes those to worker nodes.

- worker node may chop its work into yet small pieces and redistribute again

MapReduce

Reduce Step:

- master node then takes the answers to all the sub-problems and combines them in a way to get the output

MapReduce

Problem: Given an email how do you tell if it is spam?

- Count occurrences of certain words. If they occur too frequently the email is spam.

MapReduce

```
 email = ['the', 'this', 'annoy', 'the', 'the', 'annoy']
>>> def inEmail (x):
        if (x == "the"):
                return 1;
        else:
                return 0;

>>> map (inEmail, l)
[1, 0, 0, 0, 1, 1, 0]

>>> reduce ((lambda x, xs: x + xs), map(inEmail, email))
3
```

python™

Stack

- Information found here:
  - http://docs.python.org/py3k/library/stdtypes.html
- Really easy!
- Use lists to simulate a stack
- .append(value) in order to “push” onto the stack
- .pop() in order to ‘pop’ off the stack.
- Check this example out!

Queue

- Really easy too!!!!
- Use lists to simulate a queue
- .insert(0, value) in order to ‘add’ to the queue.
- .pop() in order to ‘remove’ off from the queue.
- Why does this work? What do insert and pop really do?

Queue

- Ok... this is dumb... and inefficient.
- Using lists, we get O(N), but.....
- *from collections import deque*
- *queue = deque*(['blah', 'blah', 'blah'])
- Use append() to 'add' to the queue
- Use popleft() to 'remove' from the queue
- This works in O(1)!
    - Sweeeeeeeeeet.

Dictionary

- Equivalent to Java's Map.
- Stores keys and values.
- Empty dictionary is defined by:
  - m = {}
- Pre-populated dictionary: (Map<String, String>)
  - (mapping names to SCII division…)
  - m2 = { 'jordan' : 'master', 'roy' : 'bronze', 'marty' : 'bronze' }
- Add to dictionary:
  - m2['IMMVP'] = 'master'
- Retrieve from dictionary
  - m2['jordan'] # 'master'

Dictionary

- How to get keySet()
  - map.keys()
  - Uh oh……. We get something called dict_keys…
  - We should cast to a list.
    - list(map.keys())
  - How to check if a key is contained in a dictionary…
    - ‘marty’ in m # returns true bc ‘marty’ is in dictionary
    - ‘bronze’ in m # returns false bc ‘bronze’ is not a key
  - How to delete
    - del m[‘marty’]
  - Get values with: m.values()
    - Note that you get the dict_values. So cast to a list.

Set

- s = { ‘GLaDOS’, ‘Cloud’, ‘Aeris’, ‘Shepard’, ‘Cloud’, ‘Aeris’ }
  - # {‘GLaDOS’, ‘Cloud’, “Aeris’, ‘Shepard’ }
    - Removed duplicates
- s2 = set(‘hello I am soooo happy!!!’)
  - Creates a set out of all the characters
  - The stuff in the blue must be iterable.
    - i.e. lists, tuples, etc…
- Check for contains is same as dictionary.
  - ‘GLaDOS’ in s

Set

- Tons of extra functionality
  - Check for subsets, see if set A is contained in set B
  - Go here to see more: (under 5.7)
    - http://docs.python.org/py3k/library/stdtypes.html#set.issubset

List Comprehensions

```
[expression for element in list]
```

- Applies the expression to each element in the list
- You can have 0 or more for or if statements
- If the expression evaluates to a tuple it must be in parenthesis

List Comprehensions

```
>>> vec = [2, 4, 6]
>>> [3*x for x in vec]
    [6, 12, 18]

>>> [3*x for x in vec if x > 3]
    [12, 18]

>>> [3*x for x in vec if x < 2]
    []

>>> [[x,x**2] for x in range(10)]
    [[0, 0], [1, 1], [2, 4], [3, 9]]

>>> [x, x**2 for x in vec]
    # error - parens required for tuples
```

List Comprehensions

You can do most things that you can do with map, filter and reduce more nicely with list comprehensions

How can we find out how many times ‘a’ appears in the list named email?

```
1 >>> email = ['once', 'upon', 'a', 'time', 'in', 'a',
2 'far', 'away']
3
4 >>> len( [1 for x in email if x == 'a'] )
5
6 >>> 2
```