In [1]:
```
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
```

In [78]:
```
!python --version
```

```
Python 2.7.16 :: Anaconda, Inc.
```

```
Week1: Python programming basics
  ● Hint for coding interview
  ● Important concept of python programming
  ● Data structure: Dictionary and Set
```

Python Programming Basics

(a)Hint for coding interview

```
1, what coding testing is really about?
   a, The ability of solving a problem by coding/programming
   b, Communication with the interviewer
```

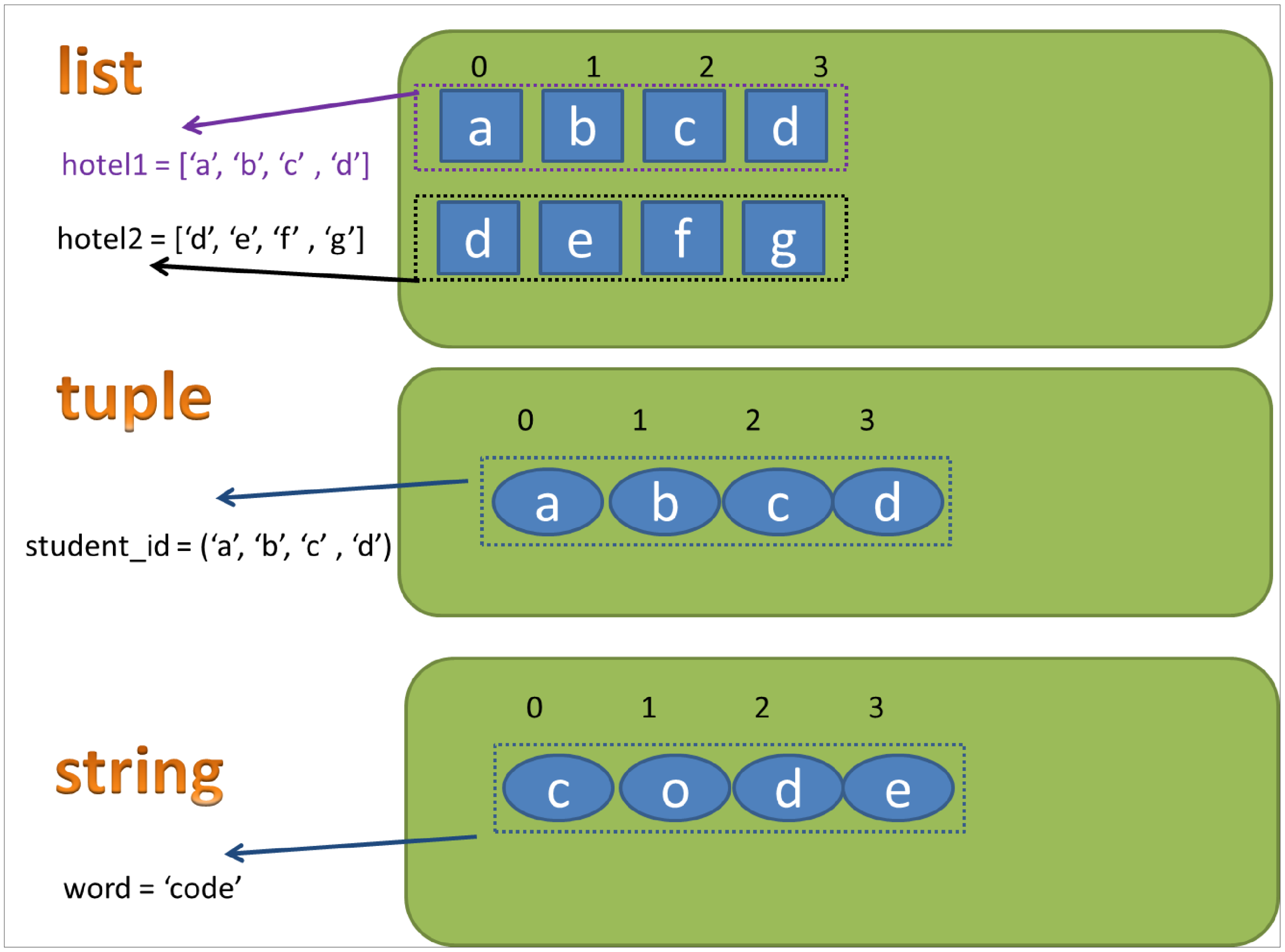
(b)Important concept of python programming

```
  ● Indentation & styling

    learn how to find good code and coding style.

    eg: https://scikit-
learn.org/stable/modules/generated/sklearn.preprocessing.normalize.html

    Be consistent!

  ● Notes on Python Variables:

    A variable name must start with a letter or the underscore character
    A variable name cannot start with a number
    A variable name can only contain alpha-numeric characters and underscores
(A-z, 0-9, and _ )
    Variable names are case-sensitive (age, Age and AGE are three different
variables)

    ex:
    my_list = [1, 2, 3]
    myList = [1, 2, 3]
    DO NOT DO THIS: a = [1, 2], b = [2, 3]
```

(c) Python data structure

```
In [3]: def show_img(filename, size_inch):
            img = plt.imread(filename)
            plt.rcParams['figure.figsize'] = [size_inch, size_inch]
            plt.imshow(img)
            plt.xticks([])
            plt.yticks([])
```

```
In [4]: show_img('./all_img/data_structure1.png', size_inch=50)
```


list
hotel1 = ['a', 'b', 'c' , 'd']
hotel2 = ['d', 'e', 'f' , 'g']
0 1 2 3
a b c d
d e f g
tuple
student_id = ('a', 'b', 'c' , 'd')
0 1 2 3
a b c d
string
word = 'code'
0 1 2 3
c o d e

List

```
In [5]: #create an empty list/the length is zero.
        nums = []
```

```
In [6]: print len(nums)
```

```
0
```

```
In [13]: #leetcode tips
         def get_first_element(nums):
              if not nums: #always check the nums is not empty. This is same as if len(nu
                 return []
              else:
                 return nums[0]
```

In [14]:
```
test = get_first_element([])
print test
```
```
[]
```

In [15]:
```
if len(nums) == 0: print nums
```
```
[]
```

In [16]:
```
if nums == []: print nums
```
```
[]
```

In [17]:
```
nums = []
nums.append(1)
nums.append(2)
print "raw nums:", nums
```
```
raw nums: [1, 2]
```

In [18]:
```
nums[-1] = 3
print "change nums as :", nums
```
```
change nums as : [1, 3]
```

In [19]:
```
nums = []
nums.append(1)
nums.append(2)
print "raw nums:", nums
```
```
raw nums: [1, 2]
```

In [20]:
```
nums_deep_copy = nums[:]
nums_deep_copy[-1] = 3
```

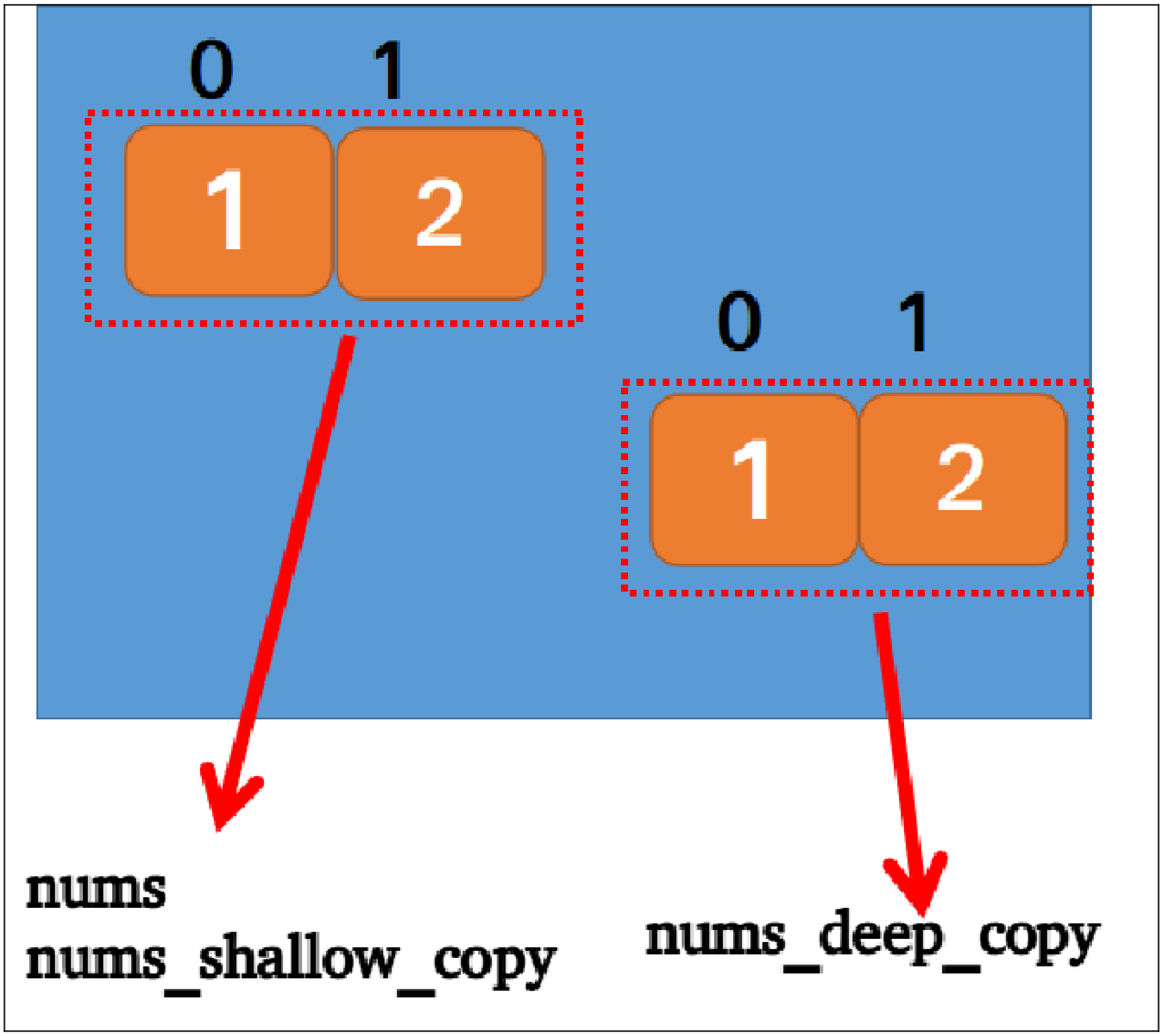
In [21]:
```
print "nums_deep_copy:", nums_deep_copy
print "the raw/old nums:", nums
```
```
nums_deep_copy: [1, 3]
the raw/old nums: [1, 2]
```

In [22]:
```
nums_shallow_copy = nums
nums_shallow_copy[-1] = 3
```

In [23]:
```
print "nums_shallow_copy:", nums_shallow_copy
print "raw nums:", nums
```
```
nums_shallow_copy: [1, 3]
raw nums: [1, 3]
```

what happend to deep copy and shallow copy?

In [24]:
```
show_img('./all_img/deep_vs_shallow.png', 12)
```



In [28]:
```
#leetcode hints & more

def get_positive1(nums):
    if not nums:
        return []
    pos_nums = []
    for e in nums:
        if e > 0:
            pos_nums.append(e)
    return pos_nums

def get_positive2(nums):
    #list comprehension
    return [e for e in nums if e > 0]

def get_positive3(nums):
    """ same get_positive function without indentation in conditions"""
    if not nums: return []
    pos_nums = []
    for e in nums:
        if e > 0: pos_nums.append(e)
    return pos_nums
```

In [27]:
```
[e for e in []]
```

Out[27]:
```
[]
```

Tuple

In [29]:
```
t_nums = ()#list []
t_nums.append(1)
print t_nums
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-29-7387a543281c> in <module>()
      1 t_nums = ()#list []
----> 2 t_nums.append(1)
      3 print t_nums

AttributeError: 'tuple' object has no attribute 'append'
```

In [34]:
```
t_nums = (1,2,3)
print t_nums
```

```
(1, 2, 3)
```

In [39]:
```
t_nums[1]
```

Out[39]:
```
2
```

```
In [37]: t_nums[-1]=4
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-37-46bb54f7ed44> in <module>()
----> 1 t_nums[-1]=4

TypeError: 'tuple' object does not support item assignment
```

String

```
In [40]: word = 'leetcode'
```

```
In [41]: print word[1]
```

```
e
```

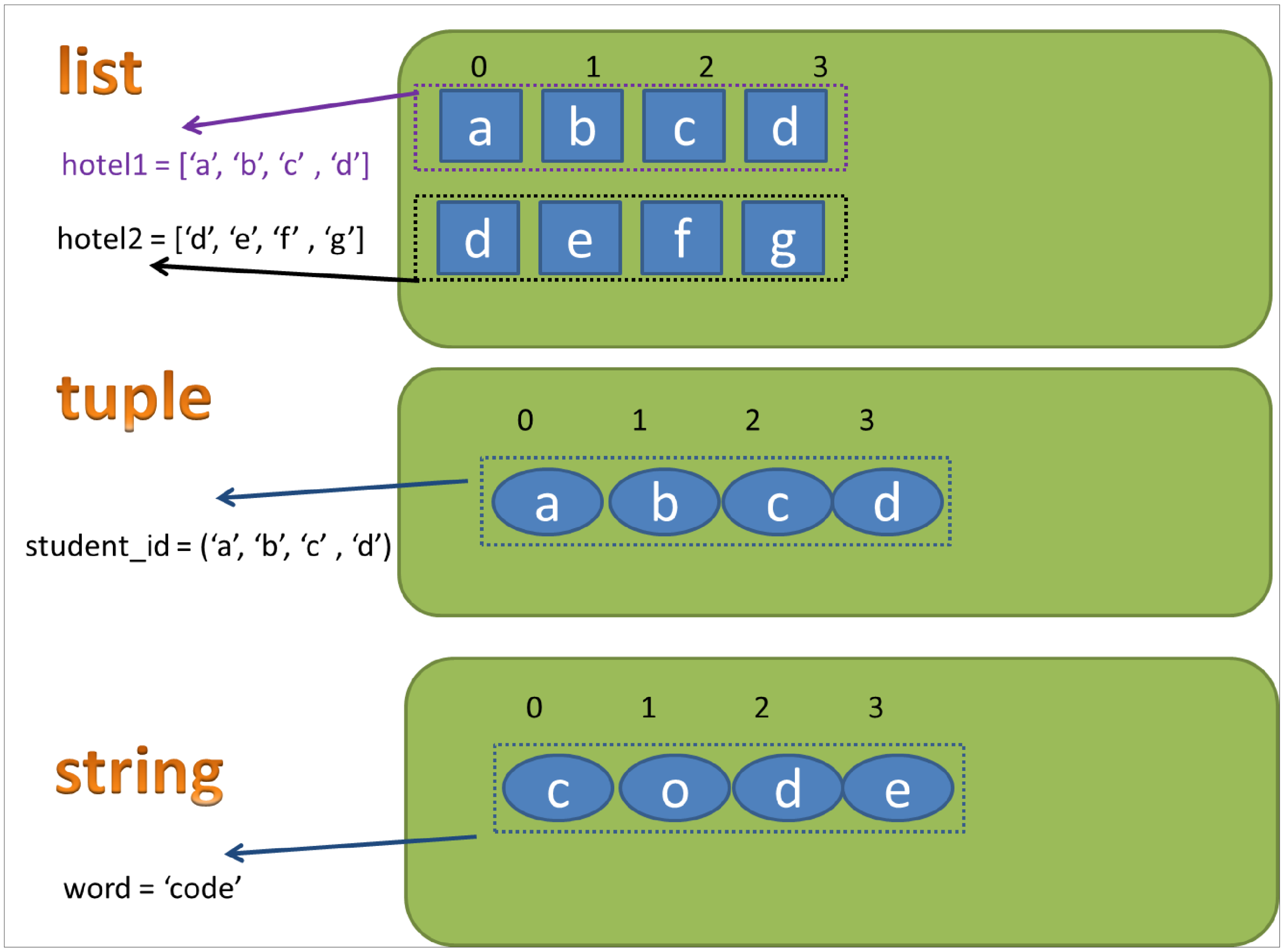
```
In [42]: word[1] = 'a'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-42-d7ab97e4ac21> in <module>()
----> 1 word[1] = 'a'

TypeError: 'str' object does not support item assignment
```

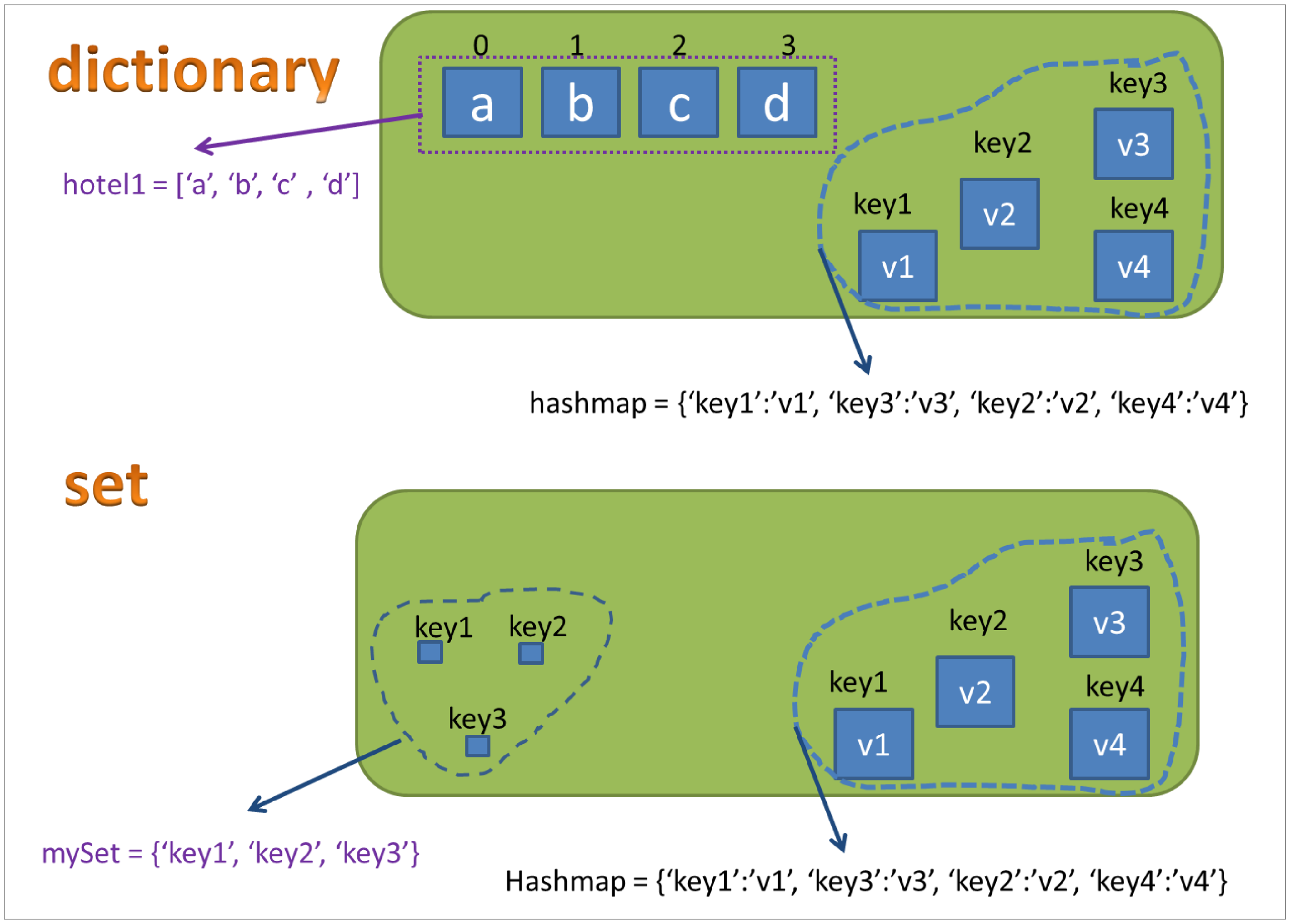
```
In [ ]: #After-class test: Type conversion.

        #list to string
        #string to list
        #list to tuple
        #tuple to list
```

Dictionary vs Set

```
In [43]: show_img('./all_img/data_structure1.png', size_inch=50)
```


list
0
1
2
3
a
b
c
d
hotel1 = [‘a’, ‘b’, ‘c’ , ‘d’]
d
e
f
g
hotel2 = [‘d’, ‘e’, ‘f’ , ‘g’]
tuple
0
1
2
3
a
b
c
d
student_id = (‘a’, ‘b’, ‘c’ , ‘d’)
string
0
1
2
3
c
o
d
e
word = ‘code’

In [44]:
```
show_img('./all_img/data_structure2.png', size_inch=50)
```


dictionary
0
1
2
3
a
b
c
d
hotel1 = ['a', 'b', 'c' , 'd']
key3
key2
key1
key4
v3
v2
v1
v4
hashmap = {'key1':'v1', 'key3':'v3', 'key2':'v2', 'key4':'v4'}
set
key1
key2
key3
mySet = {'key1', 'key2', 'key3'}
Hashmap = {'key1':'v1', 'key3':'v3', 'key2':'v2', 'key4':'v4'}

In [45]:
```
hashmap = {} #() is tuple, [] is list, {} is map, {'key1', 'key2'} is set, {'key
hashmap['key1'] = 'v1'
hashmap['key2'] = 'v2'
hashmap['key3'] = 'v3'
hashmap['key4'] = 'v4'
print hashmap
```

```
{'key3': 'v3', 'key2': 'v2', 'key1': 'v1', 'key4': 'v4'}
```

In [49]:
```
mySet = {'key1', 'key2', 'key3'}
print mySet
```

```
set(['key3', 'key2', 'key1'])
```

Time complexity preview (will cover more and more in future classes)

In [50]:
```
'key1' in hashmap #O(1) time complexity, that is why we use dictionary for look
```

Out[50]:
```
True
```

In [51]:
```
hotel1 = ['a', 'b', 'c', 'd']
'd' in hotel1    #O(n) time complexity
```

Out[51]:
```
True
```

```
In [53]: for key in hashmap:
             print key, hashmap[key]
```

```
key3 v3
key2 v2
key1 v1
key4 v4
```

```
In [54]: hashmap = {}
         two_keys = [1,2]
         hashmap[two_keys] = 'ab'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-54-b65f43ea9509> in <module>()
      1 hashmap = {}
      2 two_keys = [1,2]
----> 3 hashmap[two_keys] = 'ab'

TypeError: unhashable type: 'list'
```

```
In [55]: hashmap = {}
         two_keys = (1,2)
         hashmap[two_keys] = 'ab'
         print hashmap
```

```
{(1, 2): 'ab'}
```

```
In [56]: oneSet = {0}
         oneSet.add(1)
         oneSet.add(2)
         print oneSet, type(oneSet)
```

```
set([0, 1, 2]) <type 'set'>
```

```
In [57]: anotherSet = {0,3,4}
```

```
In [58]: oneSet&anotherSet#& == and, | or
```

```
Out[58]: {0}
```

```
In [59]: oneSet|anotherSet
```

```
Out[59]: {0, 1, 2, 3, 4}
```

Leetcode 136 Single Number

```
Given a non-empty array of integers, every element appears twice except for
one. Find that single one.

Note:

Your algorithm should have a linear runtime complexity. Could you implement it
without using extra memory?
```

```
Example 1:

Input: [2,2,1]
Output: 1
```

In [76]:
```
class Solution(object):
    def singleNumber(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        """
        new_dict = {}
        for e in nums:#for i in range(len(nums)): nums[i]
            if e not in new_dict:
                new_dict[e] = 1
            else:
                new_dict[e] +=1
        for e in nums:
            if new_dict[e] == 1:
                return e
```

In [77]:
```
p136 = Solution()
output = p136.singleNumber([2,2,1])
print "Output:", output
```

```
Output: 1
```

Leetcode 169 Majority Element

```
Given an array of size n, find the majority element. The majority element is
the element that appears more than
⌊ n/2 ⌋ times.

You may assume that the array is non-empty and the majority element always
exist in the array.

Example 1:

Input: [3,2,3]
Output: 3
```

In [ ]:
```
class Solution(object):
    def majorityElement(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        """
        length = len(nums)
        if length == 0:
            return 0
        checkLen = int(length/2)
        newDict = {}
        for e in nums:
            if e not in newDict:
                newDict[e] = 1
            else:
                newDict[e] += 1
        rkey = 0
        for key in newDict:
            if newDict[key] > checkLen:
                rkey = key
        return rkey
```

In [ ]:
```
p169 = Solution()
output = p169.majorityElement([3,2,3])
print "Output:", output
```

Leetcode 217 Contains Duplicate

```
Given an array of integers, find if the array contains any duplicates.

Your function should return true if any value appears at least twice in the
array, and it should return false if every element is distinct.

Example 1:

Input: [1,2,3,1]
Output: true
```

In [67]:
```
class Solution(object):
    def containsDuplicate(self, nums):
        """
        :type nums: List[int]
        :rtype: bool
        """
        #len(set(nums))
        #len(nums)
        return len(set(nums)) != len(nums)
```

In [66]:
```
len(set([1,1,1,2,2,3])), len([1,1,1,2,2,3])
```

Out[66]:
```
(3, 6)
```

In [ ]:
```
#leetcode hits
# newDict = {}
# for e in nums:
#     if e not in newDict:
#         newDict[e] = 1
#     else:
#         newDict[e] += 1
```

Leetcode 1 Two Sum

```
Given an array of integers, return indices of the two numbers such that they
add up to a specific target.

You may assume that each input would have exactly one solution, and you may
not use the same element twice.

Example:

Given nums = [2, 7, 11, 15], target = 9,

Because nums[0] + nums[1] = 2 + 7 = 9,
return [0, 1]
```

```
Solution:
current 2 in nums, target is 9, 2 + (7) = 9, so 7 must be in nums, namely,
current,  target - current both must be in the num.

for val in nums: 2
    if target - val in nums: 7
        output index_of_val, index_of (target-val)
```

In [68]:
```
class Solution(object):
    def twoSum(self, nums, target):
        """
        :type nums: List[int]
        :type target: int
        :rtype: List[int]
        """
        dic = {}
        for index, val in enumerate(nums):
            dic[val] = index
        #2, 1, 3, target 4
        for index, val in enumerate(nums):
            if target-val in dic:
                if index == dic[target-val]:
                    continue
                else:
                    if index <= dic[target-val]:
                        return [index, dic[target-val]]
                    else:
                        return [dic[target-val], index]
```

```
In [69]: p1= Solution()
         output = p1.twoSum([2, 7, 11, 15], 9)
         print "Output:", output

         Output: [0, 1]
```