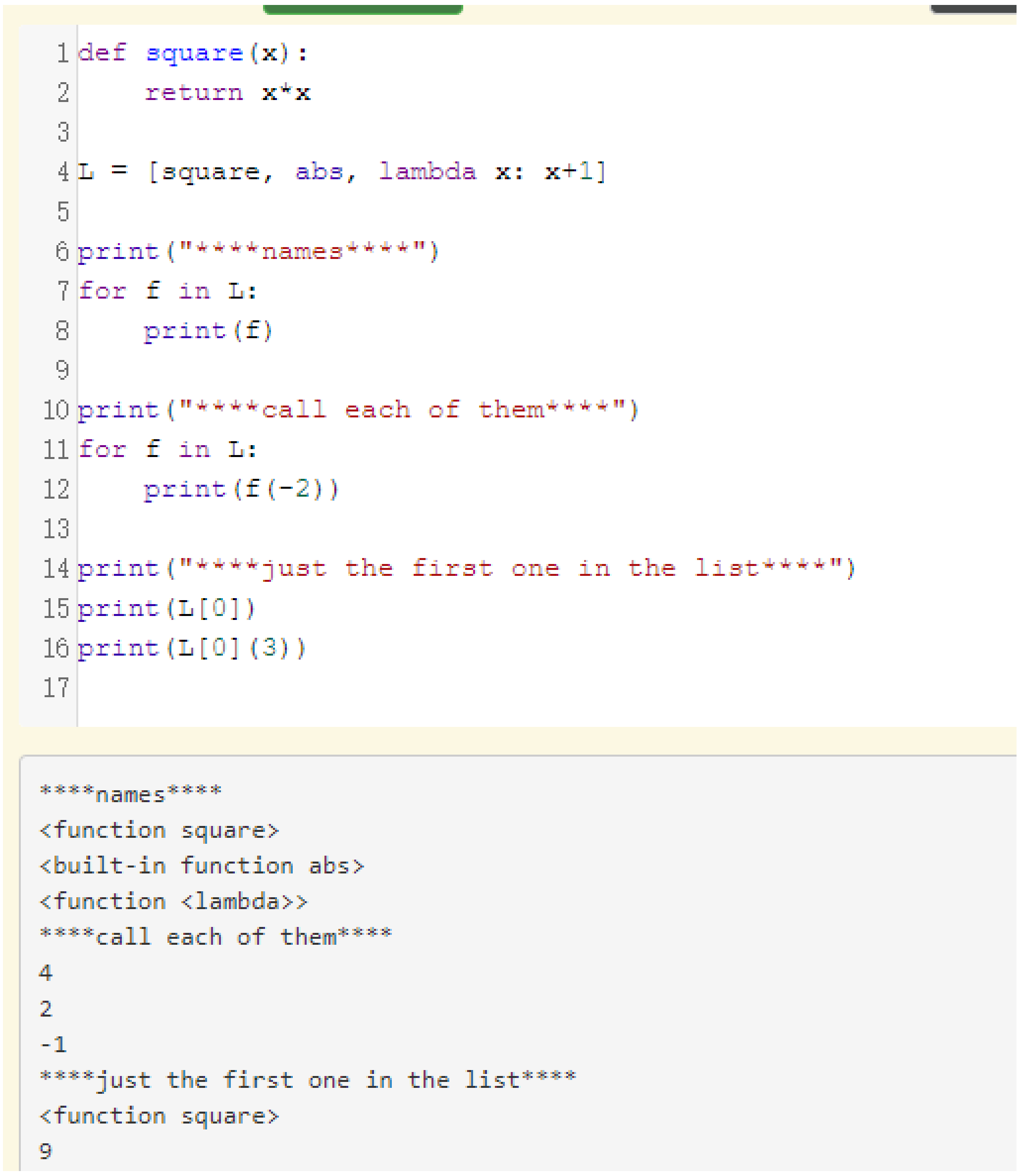
3.1 Nested Data and Nested Iteration

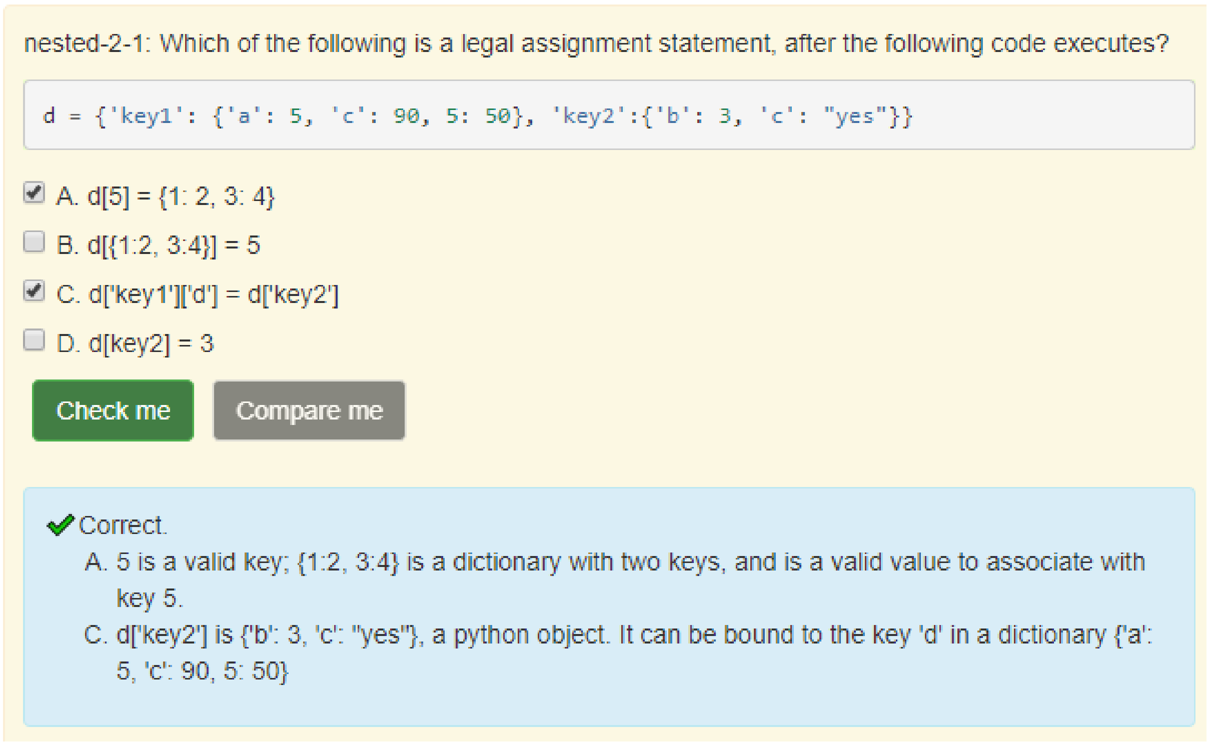
- Lists with Complex Items
  - 例子
    - nested1 = [['a', 'b', 'c'],['d', 'e'],['f', 'g', 'h']]
    - print(nested1[0])
    - print(len(nested1))
    - nested1.append(['i'])
    - print("-------")
    - for L in nested1:
    -     print(L)
    - 这种就是list里还有list的例子
  - List中List
    - nested1 = [['a', 'b', 'c'],['d', 'e'],['f', 'g', 'h']]
    - y = nested1[1]
    - print(y)
    - print(y[0])
    - print([10, 20, 30][1])
    - print(nested1[1][0])
    - 先定义y是list里面的index 1 然后print y[0]的话相当于list中list
    - 下面一行的就是print你创造的某个list中的index的某个元素 最后一行就是一个已知的list中的元素
  - List of function
    - 例子



```python
def square(x):
    return x*x

L = [square, abs, lambda x: x+1]

print("****names****")
for f in L:
    print(f)

print("****call each of them****")
for f in L:
    print(f(-2))

print("****just the first one in the list****")
print(L[0])
print(L[0](3))
```

```
****names****
<function square>
<built-in function abs>
<function <lambda>>
****call each of them****
4
2
-1
****just the first one in the list****
<function square>
9
```

- Nested Dictionary
  - 题目



nested-2-1: Which of the following is a legal assignment statement, after the following code executes?

```python
d = {'key1': {'a': 5, 'c': 90, 5: 50}, 'key2':{'b': 3, 'c': "yes"}}
```

[✓ A. d[5] = {1: 2, 3: 4}]

B. d[{1:2, 3:4}] = 5

[✓ C. d['key1']['d'] = d['key2']]

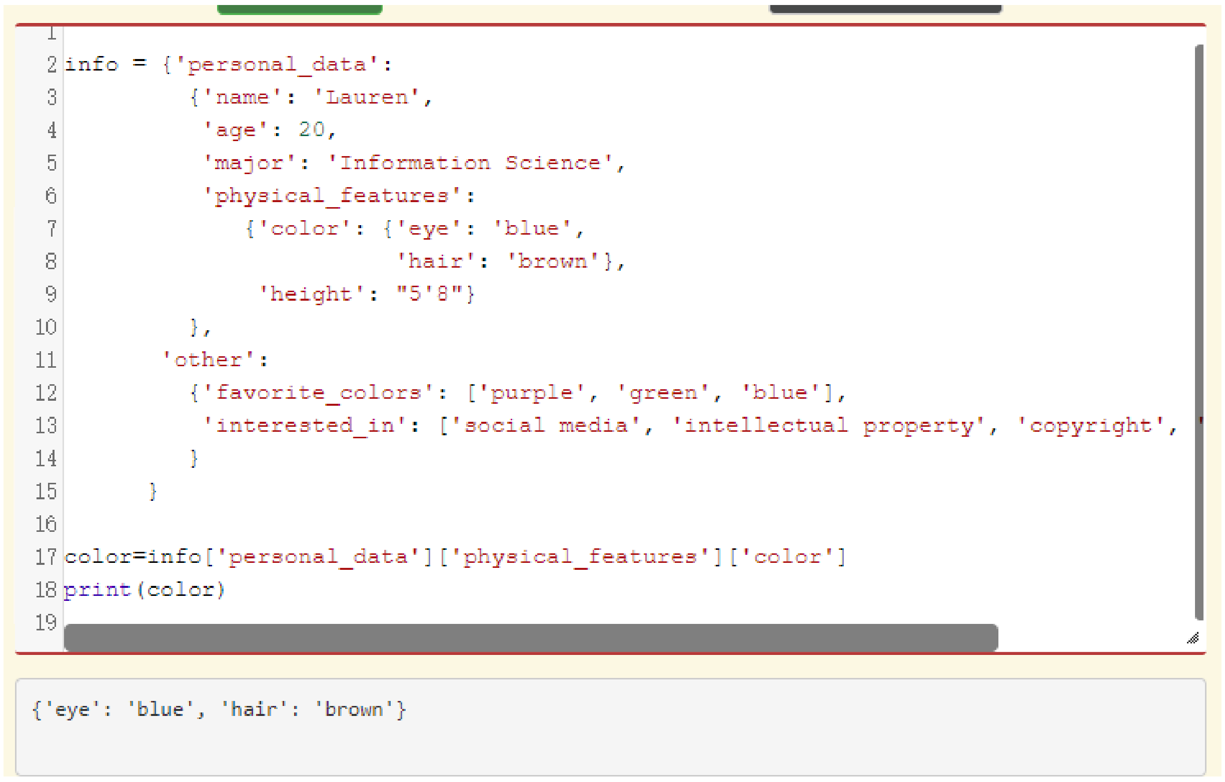
D. d[key2] = 3

✓ Correct.

A. 5 is a valid key; {1:2, 3:4} is a dictionary with two keys, and is a valid value to associate with key 5.

C. d['key2'] is {'b': 3, 'c': "yes"}, a python object. It can be bound to the key 'd' in a dictionary {'a': 5, 'c': 90, 5: 50}

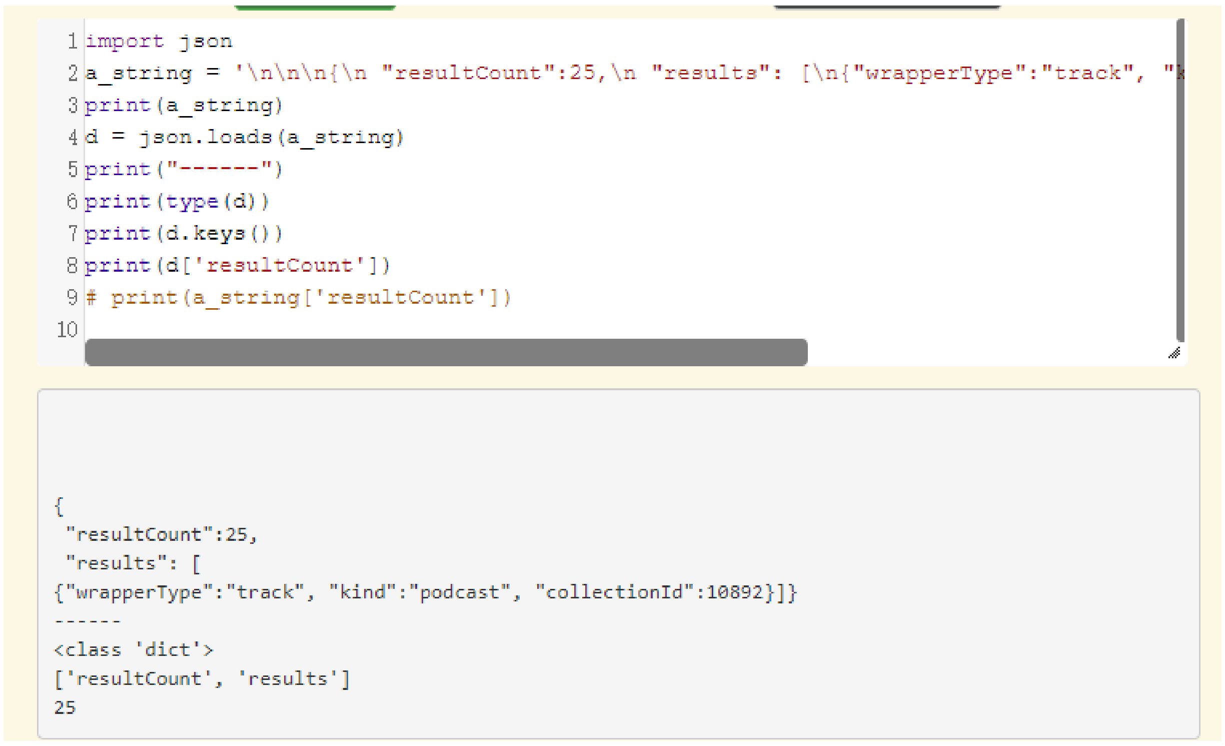
- 一定要记住带引号！得到的value还是一个dic 如果想要改变其中的变量：...=要写的东西



```
info = {'personal_data':
         {'name': 'Lauren',
          'age': 20,
          'major': 'Information Science',
          'physical_features':
             {'color': {'eye': 'blue',
                        'hair': 'brown'},
              'height': "5'8"}
         },
       'other':
         {'favorite_colors': ['purple', 'green', 'blue'],
          'interested_in': ['social media', 'intellectual property', 'copyright', '
         }
      }

color=info['personal_data']['physical_features']['color']
print(color)
```

```
{'eye': 'blue', 'hair': 'brown'}
```

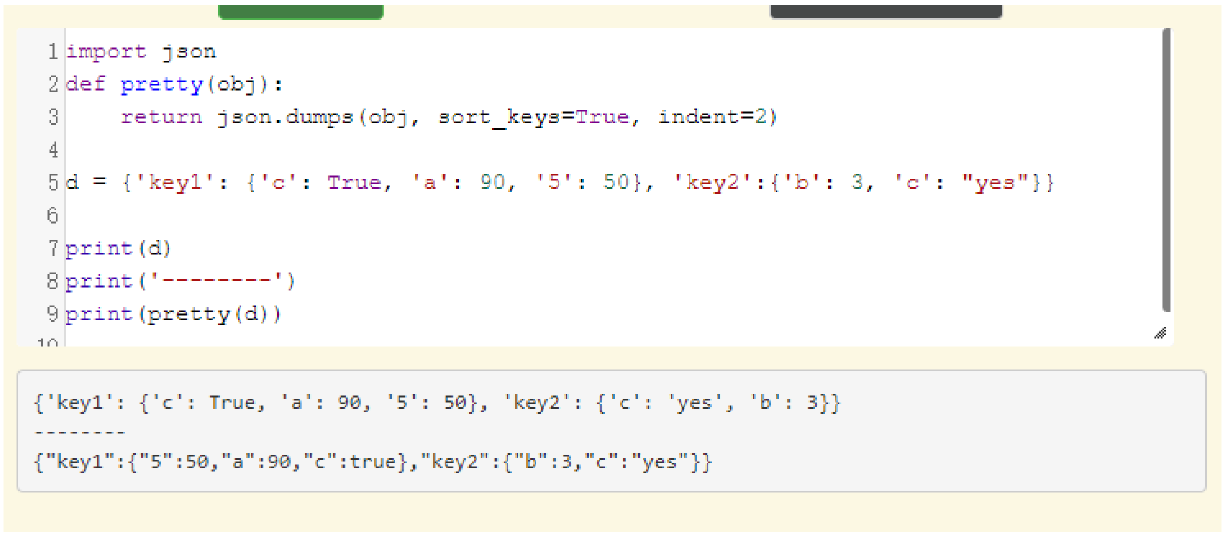
- JSON Format and JSON Module
    -
    - Javascript object notation
    - 这个module里面主要用到的就是loads和dumps两个function



```
import json
a_string = '\n\n\n{\n "resultCount":25,\n "results": [\n{"wrapperType":"track", "
print(a_string)
d = json.loads(a_string)
print("------")
print(type(d))
print(d.keys())
print(d['resultCount'])
# print(a_string['resultCount'])
```

```



{
 "resultCount":25,
 "results": [
{"wrapperType":"track", "kind":"podcast", "collectionId":10892}]}
------
<class 'dict'>
['resultCount', 'results']
25
```

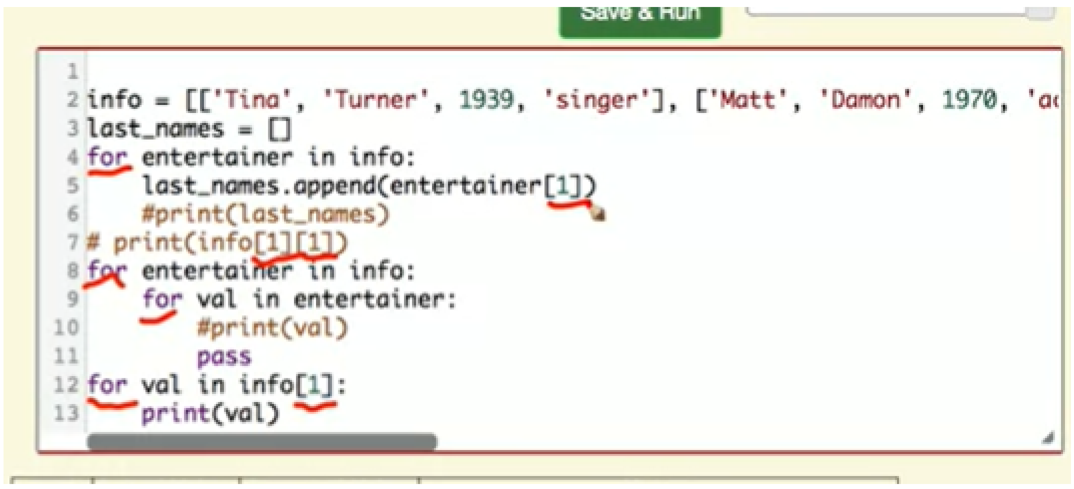
    - loads function输入string转化成dic or list   json.loads(entertainment)
    - dumps把dic or list转化成string



```
import json
def pretty(obj):
    return json.dumps(obj, sort_keys=True, indent=2)

d = {'key1': {'c': True, 'a': 90, '5': 50}, 'key2':{'b': 3, 'c': "yes"}}

print(d)
print('--------')
print(pretty(d))
```

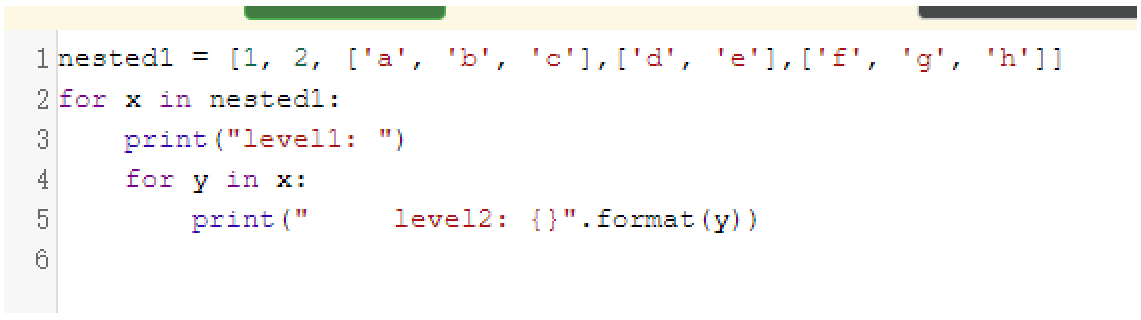
```
{'key1': {'c': True, 'a': 90, '5': 50}, 'key2': {'c': 'yes', 'b': 3}}
--------
{"key1":{"5":50,"a":90,"c":true},"key2":{"b":3,"c":"yes"}}
```

- Nested Iteration
  - 可以用2个for loop，一个for加index， 两个index[][]



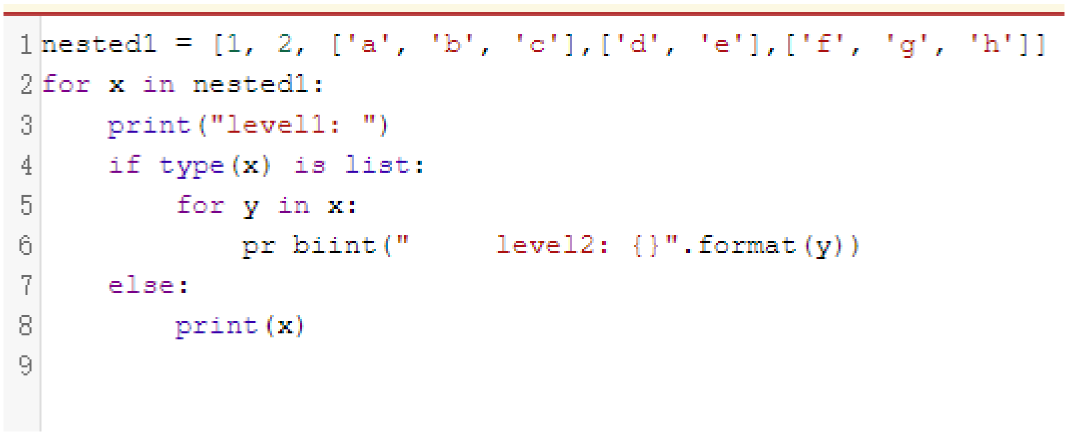
```

info = [['Tina', 'Turner', 1939, 'singer'], ['Matt', 'Damon', 1970, 'ac
last_names = []
for entertainer in info:
    last_names.append(entertainer[1])
    #print(last_names)
# print(info[1][1])
for entertainer in info:
    for val in entertainer:
        #print(val)
        pass
for val in info[1]:
    print(val)
```

  - 还有一个例子
    - 如果用for lst in L:
    - for word in lst:
    - for char in word:
    - if char=='b':
    - b_strings.append(word)这样的话如果单词有两个b就不算了
    - 所以最简单的就是 直接for之后加if'b' in word:
- Structuring Nested Data
  - 例子
    - 会报错 因为data中有int格式的1和2 但是这些是不能用for loop去iterate的



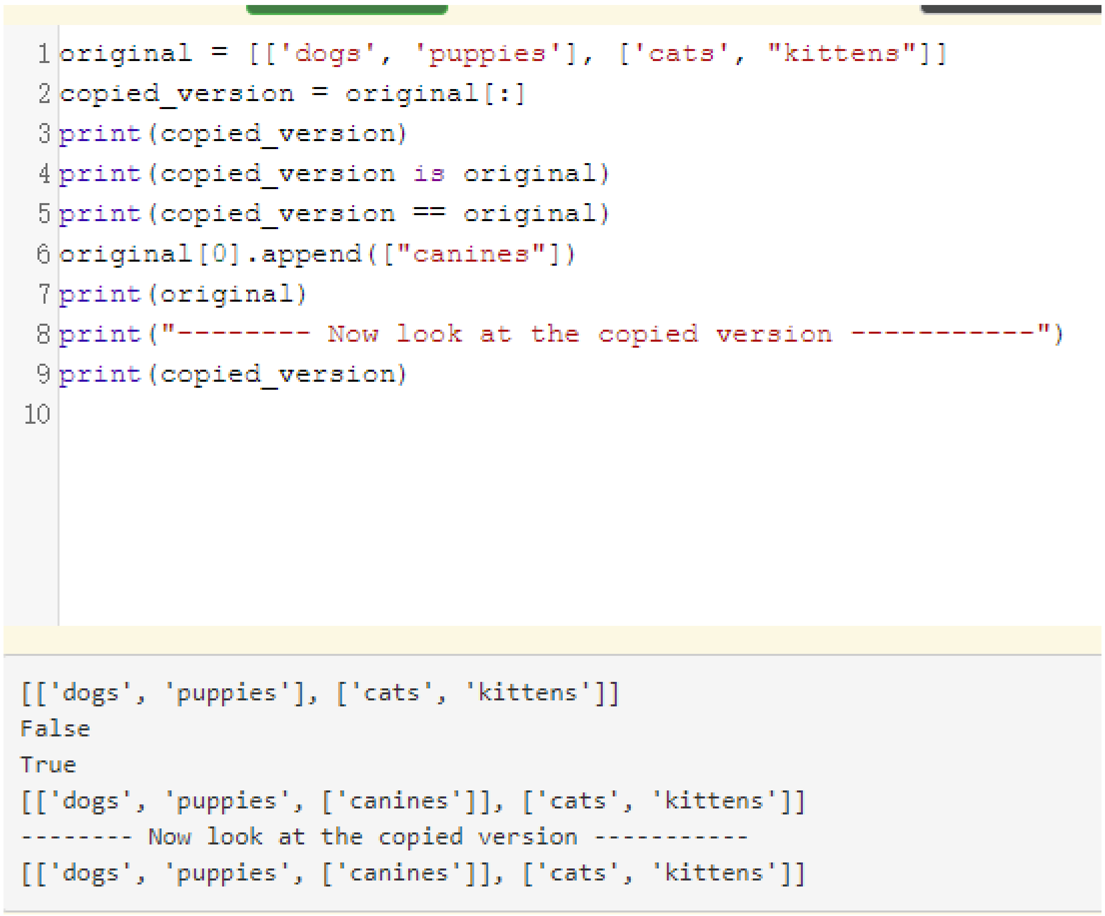
```
nested1 = [1, 2, ['a', 'b', 'c'],['d', 'e'],['f', 'g', 'h']]
for x in nested1:
    print("level1: ")
    for y in x:
        print("     level2: {}".format(y))

```

- 所以要用一个方法去让他可以 用if 如果发现type是list的就for 如果不是就print 变成



```python
nested1 = [1, 2, ['a', 'b', 'c'],['d', 'e'],['f', 'g', 'h']]
for x in nested1:
    print("level1: ")
    if type(x) is list:
        for y in x:
            pr biint("     level2: {}".format(y))
    else:
        print(x)

```

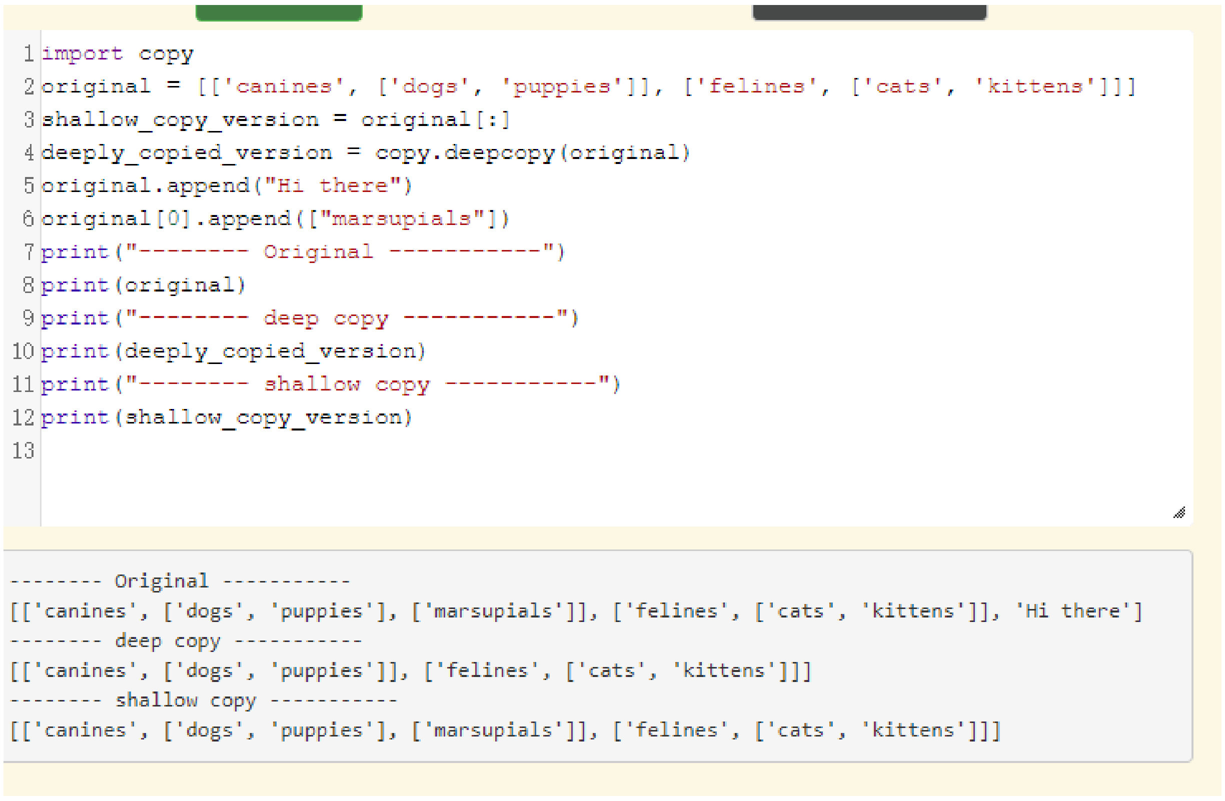
- Shallow copies 一个变其他也变
  - [:]就是从头到尾都要找到 这里就算弄了一个copied version 但因为是refer to相同的list 所以一个改变了之后另一个也随之改变了



```python
original = [['dogs', 'puppies'], ['cats', "kittens"]]
copied_version = original[:]
print(copied_version)
print(copied_version is original)
print(copied_version == original)
original[0].append(["canines"])
print(original)
print("-------- Now look at the copied version -----------")
print(copied_version)

```

```
[['dogs', 'puppies'], ['cats', 'kittens']]
False
True
[['dogs', 'puppies', ['canines']], ['cats', 'kittens']]
-------- Now look at the copied version -----------
[['dogs', 'puppies', ['canines']], ['cats', 'kittens']]
```

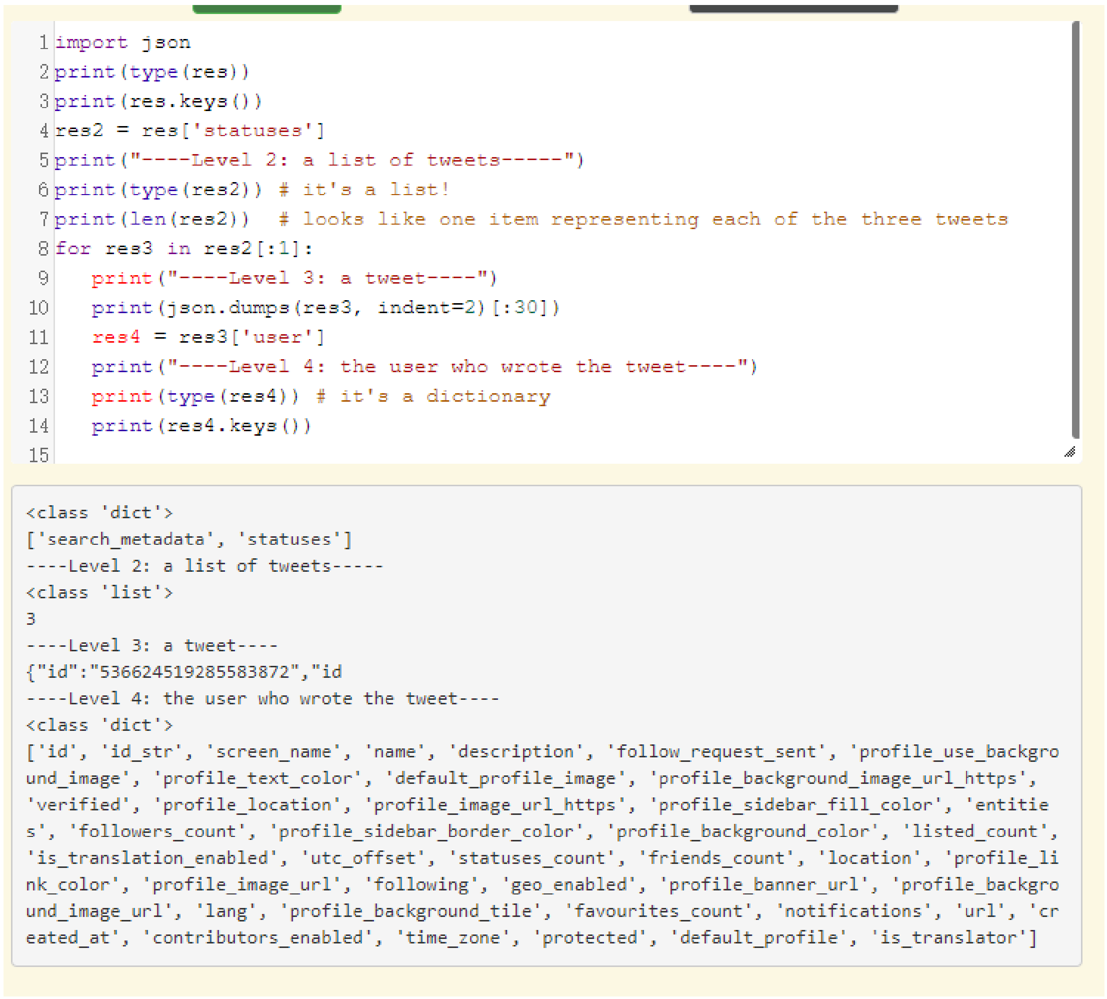
- deep copy 一个变另外的不会变
  - 例子



```python
import copy
original = [['canines', ['dogs', 'puppies']], ['felines', ['cats', 'kittens']]]
shallow_copy_version = original[:]
deeply_copied_version = copy.deepcopy(original)
original.append("Hi there")
original[0].append(["marsupials"])
print("-------- Original -----------")
print(original)
print("-------- deep copy -----------")
print(deeply_copied_version)
print("-------- shallow copy -----------")
print(shallow_copy_version)
```

```
-------- Original -----------
[['canines', ['dogs', 'puppies'], ['marsupials']], ['felines', ['cats', 'kittens']], 'Hi there']
-------- deep copy -----------
[['canines', ['dogs', 'puppies']], ['felines', ['cats', 'kittens']]]
-------- shallow copy -----------
[['canines', ['dogs', 'puppies'], ['marsupials']], ['felines', ['cats', 'kittens']]]
```

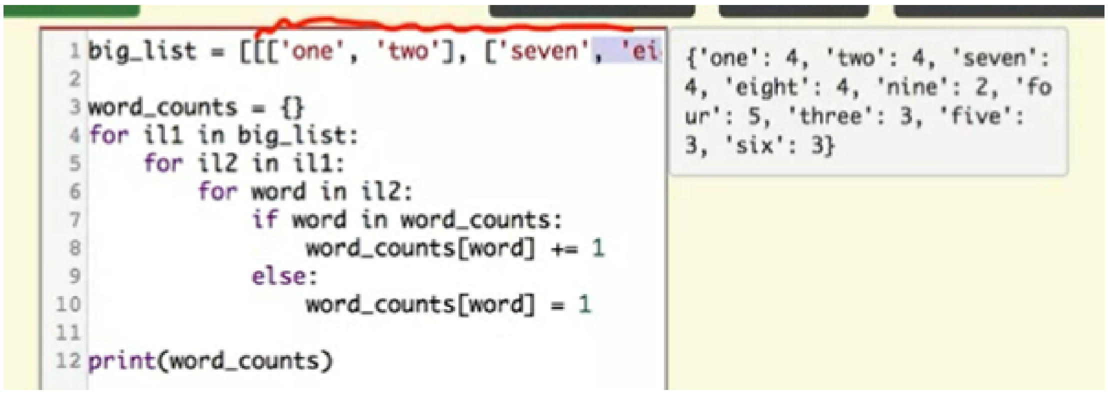
- Extracting from nested data
    - 会有很多乱七八糟的东西
        - 最开始很多一堆
            - res是个dict 然后看res里面的keys油什么 res2是个list 里面有3个元素 想找到写tweet的人 先把list用dumps转成string 读取前30个字符 然后命名res4 看看res4的type和keys



```python
import json
print(type(res))
print(res.keys())
res2 = res['statuses']
print("----Level 2: a list of tweets-----")
print(type(res2)) # it's a list!
print(len(res2))  # looks like one item representing each of the three tweets
for res3 in res2[:1]:
    print("----Level 3: a tweet----")
    print(json.dumps(res3, indent=2)[:30])
    res4 = res3['user']
    print("----Level 4: the user who wrote the tweet----")
    print(type(res4)) # it's a dictionary
    print(res4.keys())

```

```
<class 'dict'>
['search_metadata', 'statuses']
----Level 2: a list of tweets-----
<class 'list'>
3
----Level 3: a tweet----
{"id":"536624519285583872","id
----Level 4: the user who wrote the tweet----
<class 'dict'>
['id', 'id_str', 'screen_name', 'name', 'description', 'follow_request_sent', 'profile_use_backgro
und_image', 'profile_text_color', 'default_profile_image', 'profile_background_image_url_https',
'verified', 'profile_location', 'profile_image_url_https', 'profile_sidebar_fill_color', 'entitie
s', 'followers_count', 'profile_sidebar_border_color', 'profile_background_color', 'listed_count',
'is_translation_enabled', 'utc_offset', 'statuses_count', 'friends_count', 'location', 'profile_li
nk_color', 'profile_image_url', 'following', 'geo_enabled', 'profile_banner_url', 'profile_backgro
und_image_url', 'lang', 'profile_background_tile', 'favourites_count', 'notifications', 'url', 'cr
eated_at', 'contributors_enabled', 'time_zone', 'protected', 'default_profile', 'is_translator']
```

- 就看会的
    - import json
    - res2 = res['statuses']
    - for res3 in res2:
    -     res4 = res3['user']
    -     print(res4['screen_name'], res4['created_at'])
    - 输出
    - 31brooks_ Wed Apr 09 14:34:41 +0000 2014
    - froyoho Thu Jan 14 21:37:54 +0000 2010
    - MDuncan95814 Tue Sep 11 21:02:09 +0000 2012
- 或者如果想要方便且debug容易的话
- for res3 in res['statuses']:
-     print(res3['user']['screen_name'], res3['user']['created_at'])最好
- 一个例子
    - 一层层地把洋葱剥下去 然后慢慢for



```
big_list = [[['one', 'two'], ['seven', 'ei

word_counts = {}
for il1 in big_list:
    for il2 in il1:
        for word in il2:
            if word in word_counts:
                word_counts[word] += 1
            else:
                word_counts[word] = 1

print(word_counts)
```

```
{'one': 4, 'two': 4, 'seven':
4, 'eight': 4, 'nine': 2, 'fo
ur': 5, 'three': 3, 'five':
3, 'six': 3}
```