2.2 Dictionary Mechanics

- Dictionaries
  - 跟string这些差不多的一个类型 只不过是a bag of key value pairs
  - creat的时候要用{ } 符号
  - 例子
    - 1 eng2sp = {} 建立一个新字典
    - 2 eng2sp['one'] = 'uno' assign one to uno的方式
    - 3 eng2sp['two'] = 'dos'
    - 4 eng2sp['three'] = 'tres'
    - 5 print(eng2sp)
    - 输出{'one': 'uno', 'two': 'dos', 'three': 'tres'}每一对都是用逗号分开 一对的两个用冒号 如果是string形式每个里还有“”
    - 存储的是key---value这样一对pair存储
    - dictionary是 unordered 不算顺序
    - 或者直接一行写也行
      - eng2sp = {'three': 'tres', 'one': 'uno', 'two': 'dos'}
      - print(eng2sp)
    - value = eng2sp['two']
    - print(value) 要用[]去选哪个元素
    - 或者[]也可以用来加元素
      - olympics={'gold':7}
      - olympics['silver']=8
      - 另一种写法是medals={'gold':33,'silver':17,'bronze':12}直接所有
      - olympics['bronze']=6
- Dictionary operations
  -

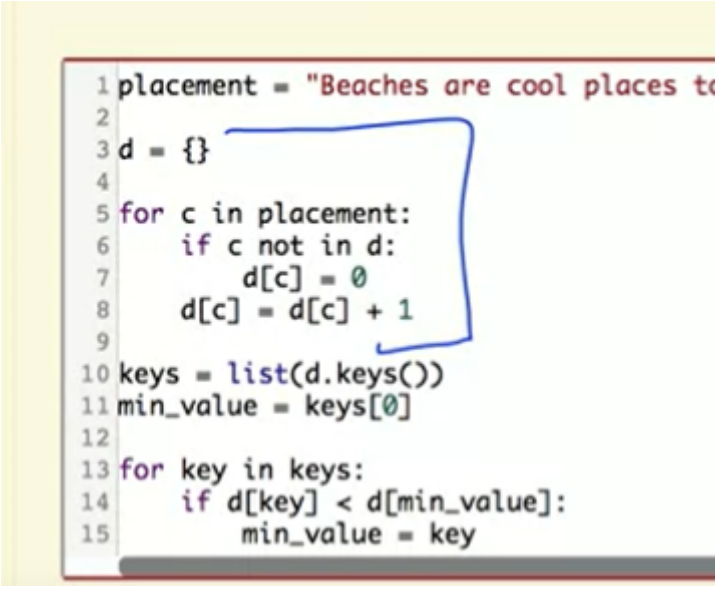
Method	Parameters	Description
keys	none	Returns a view of the keys in the dictionary
values	none	Returns a view of the values in the dictionary
items	none	Returns a view of the key-value pairs in the dictionary
get	key	Returns the value associated with key; None otherwise
get	key,alt	Returns the value associated with key; alt otherwise

- delete key value in the dictionary
  - inventory = {'apples': 430, 'bananas': 312, 'oranges': 525, 'pears': 217}
  - del inventory['pears']
- 换key 的 value
  - inventory = {'apples': 430, 'bananas': 312, 'oranges': 525, 'pears': 217}
  - inventory['pears'] = 0
  - 还有比如说inventory['bananas'] = inventory['bananas'] + 200 更新banana的值为512
  - numItems = len(inventory) 对dictionary去len方程的话是数有多少对在dictionary里
- 很多keys
  - inventory = {'apples': 430, 'bananas': 312, 'oranges': 525, 'pears': 217}
  - for key in inventory.keys()
  - print(key)
  - 输出 apples bananas oranges pears
- 如果想要keys的一个list
  - keys =list(inventory.keys()) 如果只写inventory.keys()的话其实得到的不是一个list 所以需要前面再加个list
  - print (keys)
- 如果想要apples和它代表的值一起出现的话
  - print(key,"has the value",inventory[key])
- 想要value的综合
  - Junior = {'SI 206':4, 'SI 310':4, 'BL 300':3, 'TO 313':3, 'BCOM 350':1, 'MO 300':3}
  - credits=0
  - for module in Junior:
  - credits=credits+Junior[module]
- 如果想要inventory里的数字的话
  - print(list(inventory.valus())
- 如果想要一个所有pairs的list 里面的是tuple的形式
  - print(list(inventory.items())
  - 输出[("apples",430),.....]
- 如果 print("apples" in inventory) 就会输出boolean 要么True 要么False
  - mydict = {"cat":12, "dog":6, "elephant":23, "bear":20} print(23 in mydict)就不行 因为这个operate只在keys有的情况下说True
- 另一个方法是.get
  - 很像 indexing 是用来索引的
  - inventory = {'apples': 430, 'bananas': 312, 'oranges': 525, 'pears': 217}

    - pring(inventory.get("apples"))跟inventory['apples']一样
    - 但是和index不一样的是 如果.get没找到值的话输出的是None index会直接报错 print(inventory['cherries'])
    - print(inventory.get("cherries",0)) 这个意思就是说 如果cherries在dic里面的话就print它的值 如果不在 赋值给cherries 为0
  - aliasing of dictionary 同样成立
- Dictionary Accumulation
  - 比如说现在有个task 要知道某txt中到底有几个字母t
  - 最一般的写法是
    - f = open('scarlet.txt', 'r')
    - txt = f.read()
    - t_count = 0
    - for c in txt:
    - if c == 't':
    - t_count = t_count + 1
    - print("t: " + str(t_count) + " occurrences")
    - 如果c=='t' True的话就加一
  - 但是这个情况只限制于找一个letter
  - 如果要找多的话很有可能就很麻烦了
    - f = open('scarlet.txt', 'r')
    - txt = f.read()
    - t_count = 0
    - s_count = 0
    - for c in txt:
    - if c == 't':
    - t_count = t_count + 1
    - elif c == 's':
    - s_count = s_count + 1
    - print("t: " + str(t_count) + " occurrences")
    - print("s: " + str(s_count) + " occurrences")
  - 所以这里的话就想个办法 看是不是用dic能简便
    - f = open('scarlet.txt', 'r')
    - txt = f.read()
    - x = {} 先建个空的dic
    - x['t'] = 0 给dic里面的字母t设定值0

- x['s'] = 0 同样给s设定0
- for c in txt:
- if c == 't':
- x[c] = x[c] + 1
- elif c == 's':
- x[c] = x[c] + 1
- print("t: " + str(x['t']) + " occurrences")
- print("s: " + str(x['s']) + " occurrences")

- 但是这样在碰到很多字母的时候还是要写很多elir 所以又有了下面的简化版
  - f = open('scarlet.txt', 'r')
  - txt = f.read()
  - x = {}
  - for c in txt:
  - if c not in x:
  - x[c] = 0 如果没见过就assign为0
  - x[c] = x[c] + 1 不论见没见过都加一
  - print("t: " + str(x['t']) + " occurrences")
  - print("s: " + str(x['s']) + " occurrences")
  - 或者print的时候直接print(x+":"+str(x[c])+"occurrences")
  - 或者直接用一个for loop去print
- 一个应用 scrabble game?
  - f = open('scarlet2.txt', 'r')
  - txt = f.read()
  - x = {}
  - for c in txt:
  - if c not in x:
  - x[c] = 0
  - x[c] = x[c] + 1
  - letter_values = {'a': 1, 'b': 3, 'c': 3, 'd': 2, 'e': 1, 'f':4, 'g': 2, 'h':4, 'i':1, 'j':8, 'k':5, 'l':1, 'm':3, 'n':1, 'o':1, 'p':3, 'q':10, 'r':1, 's':1, 't':1, 'u':1, 'v':8, 'w':4, 'x':8, 'y':4, 'z':10}
  - tot = 0
  - for y in x:
  - if y in letter_values:
  - tot = tot + letter_values[y] * x[y]
  - print(tot)

- 一个大应用：数清楚dic里有多少个字母之后 再找到最小数字对应的key
  - 

```
placement = "Beaches are cool places to

d = {}

for c in placement:
    if c not in d:
        d[c] = 0
    d[c] = d[c] + 1

keys = list(d.keys())
min_value = keys[0]

for key in keys:
    if d[key] < d[min_value]:
        min_value = key
```

  -