5.2 字典

字典是一个用“键”做索引来存储数据的集合。一个键和它所对应的数据形成字典中的一个条目。

- 创建字典

用花括号{ } 来表示，每个元素用冒号分隔 键和数据。

students = {3180101:’张三’, 3180102:’李四’, 3180105:’王五’, 3180110:’赵六’}

- 可以用{}或者dict()来创建空字典。

用dict()创建字典

- >>>fac=dict([("math","0001"),("python","0002"),("c","0003")])
- >>> fac
- {'math': '0001', 'python': '0002', 'c': '0003'}
- >>> fac=dict(math="0001",python="0002",c="0003")
- >>> fac
- {'math': '0001', 'python': '0002', 'c': '0003'}
- 注意：后一种方式要用标识符

字典的键

- 不可变对象可作为字典的键，如

  数字，字符串，元组

  可变对象不可以作为字典的键，如：

  列表，字典等

字典的基本运算

- 访问和修改条目

直接用[ ]运算符，用<字典> [键]的形式，访问键所对应的数据。

```
score = {'张三':78, '李四':92}
print(score['张三'])      #访问score中键为'张三'的数据
score['李四'] = 89        #把score中键为'李四'的数据修改为89
score['王五'] = 100       #score中没有键为‘王五’的元素，则增加一项，键为’王五’,数据为100。
print(score）
```

输出：

```
78
{'张三': 78, '李四': 89, '王五': 100}
```

字典的基本运算（续）

- 删除条目

用del语句，删除指定键的字典条目。

```
del score['张三']
```

注：如果指定键不存在，则会抛出KeyError异常。

字典的基本运算（续2）

- 遍历字典
  - 使用循环语句实现

```
score = {'张三':78, '李四':92, '王五':89}
for name in score:
    print(name + ':' + str(score[name]))
```

输出：

```
张三:78
李四:92
王五:89
```

字典的基本运算（续3）

- 字典大小
  用函数len()得到字典的条目的数量。
  len(score)

- 检测
  用in和not in运算符检测一个键是否在字典中存在。
  score = {'张三':78, '李四':92, '王五':89}
  print('李四' in score)
  print('小白' in score)
  输出：
  True
  False

字典的基本运算（续4）

- 用==和!=比较2个字典是否相同（键和值都相同）

```
score = {'张三':78, '李四':92, '王五':89}
mark = {'张三':78, '李四':92, '王五':91}
print(score == mark)
```

输出：

False

字典方法或函数

表5-3：常见的字典方法

函数	返回值和说明
keys()	返回由全部的键组成的一个序列
values()	返回由全部的值组成的一个序列
items()	返回一个序列，其中的每一项是一个元组，每个元组由键和它对应的值组成
clear()	删除所有条目
get(key)	返回这个键所对应的值
pop(key)	返回这个键所对应的值，同时删除这个条目

字典方法或函数（续）

- 说明
- 函数get()和运算符[ ]不同之处，在于如果键key在字典中不存在，则get(key)返回None值，而运算符[ ] 会抛出KeyError异常。
- 函数keys()、values()和items()都是返回一个序列。由于字典中键不重复，所以keys()和items()的返回结果可以转换成元组；而values()返回值由于可能存在重复值，应该转换为列表。

```
score = {'张三':78, '李四':92, '王五':92}
v = list(score.values())
print(v)
```

```
[78, 92, 92]
```

用items()实现字典遍历

- score = {'张三':78, '李四':92, '王五':89}
- for key,value in score.items():
- 　print(key + ':' + str(value))

- 张三:78
- 李四:92
- 王五:89

用户喜欢的语言

```
favorite_languages = {
    'jen': 'python',
    'sarah': 'c',
    'edward': 'ruby',
    'phil': 'python',
    }

for name, language in favorite_languages.items():
    print(name.title() + "'s favorite language is " +
        language.title() + ".")
```

用字典代替分支语句

- 输入一个1到7的数字，输出对应的星期名的缩写

- days={1:"Mon",2:"Tue",3:"Wed",4:"Thu",5:"Fri",6:"Sat",7:"Sun"}
- num=int(input())
- print(days[num])

猜姓名的姓


叶 梁 胡 程 冯 赵 张
马 郑 高 董 徐 陈 苏
周 沈 许 魏 蒋 韩 贾
李 彭 袁 何
萧 程 冯 唐 杨 赵 张
王 阎 郭 马 郑 宋 高
朱 吴 徐 邓 许 丁 魏
蒋 卢 彭 曾
罗 梁 孙 胡 萧 程 杨
赵 郑 宋 高 朱 董 于
蔡 陈 刘 傅 沈 丁 魏
贾 彭 曾
孙 胡 萧 程 冯 唐 高
朱 董 于 吴 徐 蔡 吕
苏 周 黄 丁 魏 蒋 卢
谢 韩 贾
薛 叶 罗 梁 萧 程 冯
唐 阎 郭 马 郑 宋 董
于 谢 韩 何 林
薛 叶 阎 蔡 吕 苏 傅
沈 邓 许 丁 魏 蒋 卢
贾 彭 曾 袁 柴

猜姓名的姓的程序

- nameDict={"000001":"李","000010":"王","000011":"张","000100":"刘","000101":"陈",
- "000110":"杨","000111":"赵","001000":"黄","001001":"周","001010":"吴",
- "001011":"徐","001100":"孙","001101":"胡","001110":"朱","001111":"高",
- "010000":"林","010001":"何","010010":"郭","010011":"马","010100":"罗",
- "010101":"梁","010110":"宋","010111":"郑","011000":"谢","011001":"韩",
- "011010":"唐","011011":"冯","011100":"于","011101":"董","011110":"萧",
- "011111":"程","100000":"柴","100001":"袁","100010":"邓","100011":"许",
- "100100":"傅","100101":"沈","100110":"曾","100111":"彭","101000":"吕",
- "101001":"苏","101010":"卢","101011":"蒋","101100":"蔡","101101":"贾",
- "101110":"丁","101111":"魏","110000":"薛","110001":"叶","110010":"阎"}

- s=input("输入你的选择:")
- print (nameDict[s])
- print("a")

四则运算

```
result={"+":"x+y","-":"x-y","*":"x*y","/":"x/y if y!=0   \
        else "divided by zero""}
x=int(input())
z=input().strip()
y=int(input())
r=eval(result.get(z))
if type(r)!=str:
    print(format(r,'.2f'))
else:
    print(r)
```

用字典计数

- 输入一行字符，求字符”a”,”b”和”c”出现的次数

- diccount={char:0 for char in "abc"} #字典初始化
- s=input()
- lst=[char for char in s if
- ord("a")<=ord(char)<=ord("c")]
- for char in lst:
- diccount[char]+=1
- print(diccount)

输入一行字符，求每个字符出现的次数

- 建字典dicchar,键是字符，值是字符出现的次数。由于不能预知出现哪些字符，所以不能预先初始化字典。
- 注意字典的get函数：get() 函数返回指定键的值，如果值不在字典中返回默认值。countchar.get(c,0)函数返回键为c的值, 不在字典中返回0。

- str=input()
- countchar={}

- for c in str:
- countchar[c]=countchar.get(c,0)+1
- print(countchar)

按姓名成绩排名输出

```
>>> scores={85:"李鸣",74:"黄辉",92:"张檬",88:"于静颂",63:"钱多多"}
>>> scores
{88: '于静颂', 74: '黄辉', 92: '张檬', 85: '李鸣', 63: '钱多多'}
>>> L1=list(scores.keys())
>>> L1
[88, 74, 92, 85, 63]
>>> L1.sort(reverse=True)
>>> L1
[92, 88, 85, 74, 63]
>>> L2=[]
>>> L2.append(scores[L1[0]])
>>> L2.append(scores[L1[1]])
>>> L2.append(scores[L1[2]])
>>> L2.append(scores[L1[3]])
>>> L2.append(scores[L1[4]])
>>> L2
['张檬', '于静颂', '李鸣', '黄辉', '钱多多']
```

得到L2列表的过程在学习了循环结构后可改为：

```
>>> L2=[]
>>> for i in range(0,len(L1)):
        L2.append(scores[L1[i]])
>>> L2
```

建立乘法表字典，元组作为关键字

可以根据两个乘数，查阅字典得到乘积。

以3的乘法为例：

```
>>>
d1={(3,1):3,(3,2):6,(3,3):9,(3,4):12,(3,5):15,(3,6):18,(3,7):21,(3
,8):24,(3,9):27}
>>> d1
{(3, 8): 24, (3, 2): 6, (3, 9): 27, (3, 3): 9, (3, 6): 18, (3, 7): 21,
(3, 1): 3, (3, 4): 12, (3, 5): 15}
>>> d1[(3,9)]
27
```

字典列表嵌套

```
# Store information about a pizza being ordered.
pizza = {
    'crust': 'thick',
    'toppings': ['mushrooms', 'extra cheese'],
    }

# Summarize the order.
print("You ordered a " + pizza['crust'] + "-crust pizza " +
      "with the following toppings:")

for topping in pizza['toppings']:
    print("\t" + topping)
```

二维表格嵌套表示，求最低价

name	price	Store
C#从入门到精通	25.7	卓越
ASP.NET高级编程	44.5	卓越
Python核心编程	24.7	当当
JavaScript大全	45.7	当当
u"Django简明教程	26.7	新华书店
深入Python	55.7	新华书店

```
books=[
        {"name":u"C#从入门到精通","price":25.7,"store":u"卓越"},
        {"name":u"ASP.NET高级编程","price":44.5,"store":u"卓越"},
        {"name":u"Python核心编程","price":24.7,"store":u"当当"},
        {"name":u"JavaScript大全","price":45.7,"store":u"当当"},
        {"name":u"Django简明教程","price":26.7,"store":u"新华书店"},
        {"name":u"深入Python","price":55.7,"store":u"新华书店"},
    ]

print(min([item["price"] for item in books ]))
```

图的字典表示

```
# %% 求图的顶点数，边数，边的总长  空字典{}，空集合 set()
d={"s": {1:1,2:2,3:4},
   1:{3:5},
   2:{"t":8,3:78},
   3:{2:4,"t":8},
   "t":{}   }
v=set();    e=set()    ;s=0          #顶点的集合，边点的集合
for key,value in d.items():
    v.add(key)
    if type(value)==dict:
        for key1,value1 in value.items():
            v.add(key1)
            e.add((key,key1))
            s+=value1

print(len(v),len(e),s)
```