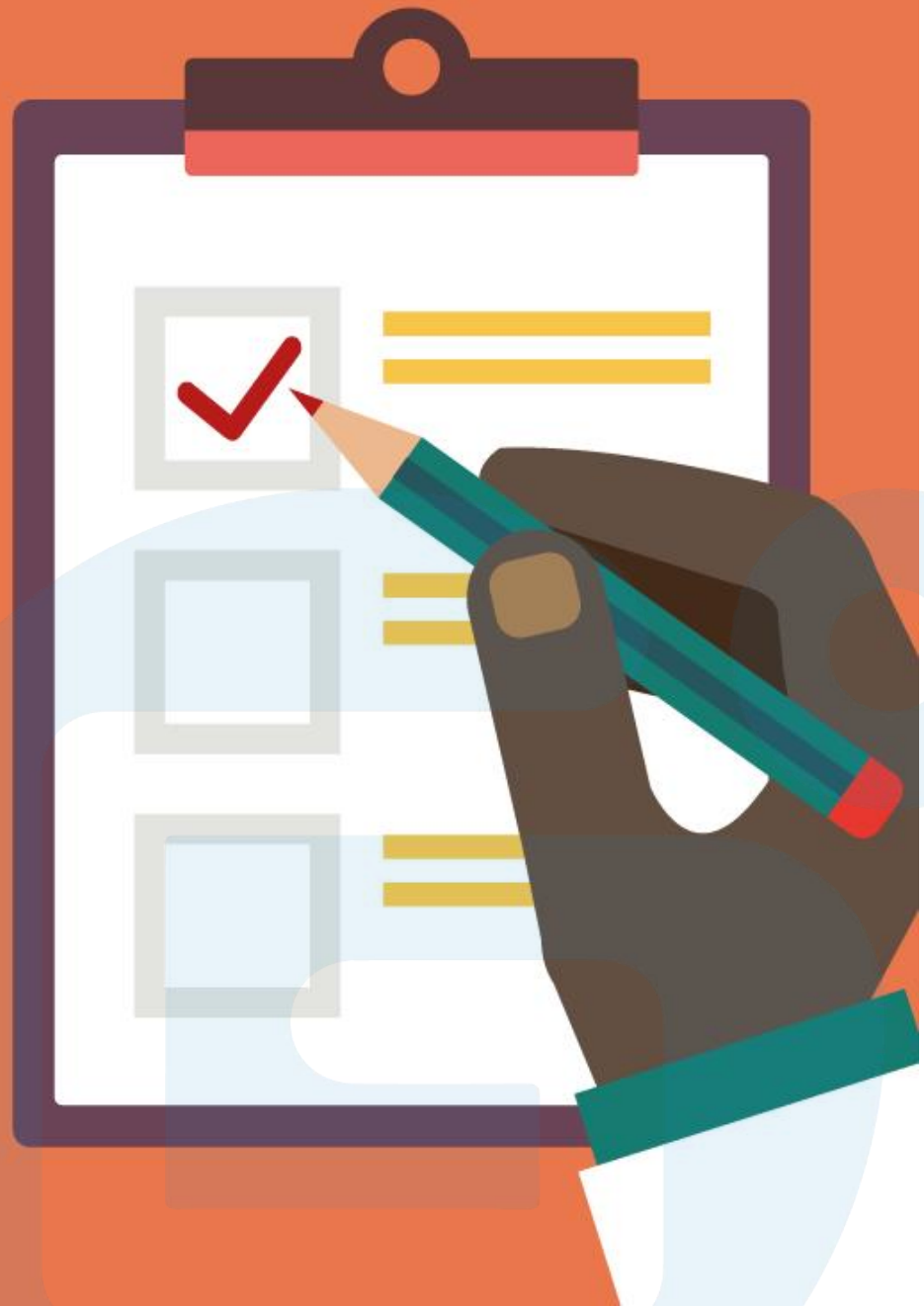
Talk is cheap, show me the code

第七课：Python4种基本容器

Python初阶入门课程系列



OUTLINE

- 容器的概念
- 序列
- 字典
- 集合
- 总结
- 实例

一　容器的概念

容器的概念

- 列表(list)
- 集合(set)
- 元组(tuple)
- 字典(dict)

Python中，可包含其他对象的对象，称之为“容器”。容器是一种数据结构。

常用的容器主要划分为两种：序列（如：列表、元祖等）和映射（如：字典）。

序列中，每个元素都有下标，它们是有序的。映射中，每个元素都有名称（又称“ 键 ”），它们是无序的。

除了序列和映射之外，还有一种需要注意的容器——“ 集合 ”。

容器的概念

定义容器

列表(list)

- 变量名称 = [元素]
- 变量名称 = list([元素])

集合(set)

- 变量名称 = {元素}
- **注意：**无序并且不能重复

元组(tuple)

- 变量名称 = （元素）
- **注意：**1. 元组是一系列固定的值（不可变数据类型）
  2. 在Python中，如果只有一个元素，并且该元素是一个数字，则需要加上一个逗号来表示，该变量是元组而不是数字

字典(dict)

- 变量名称 = {键1:值1,键2:值2}
- **注意：**字典是以键值对存在的，每个键对应一个唯一的值，键必须是字符串

容器的概念

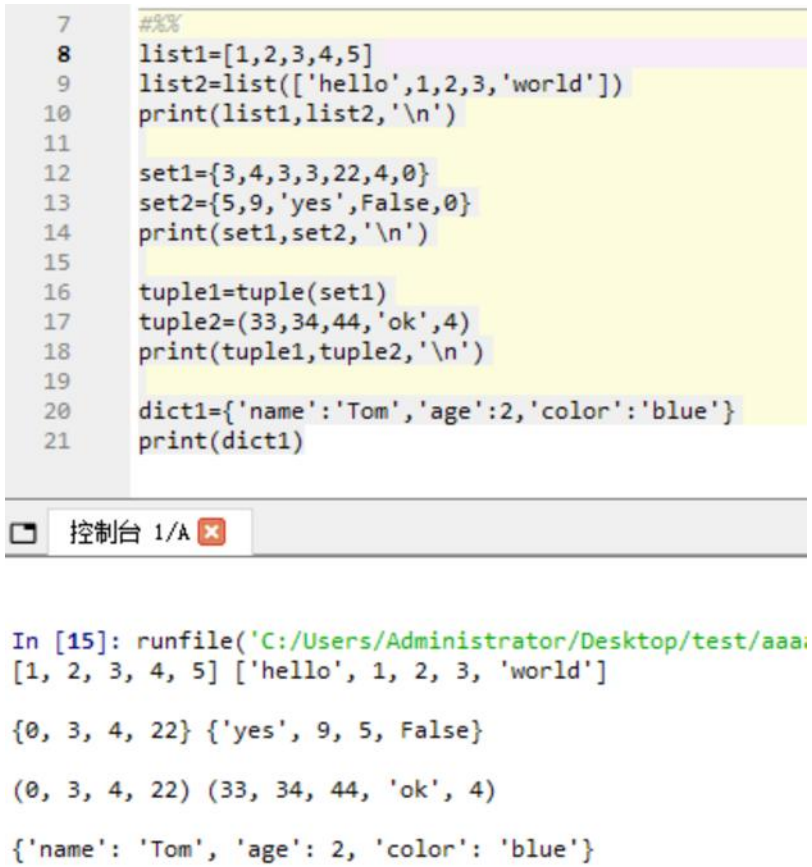
定义容器

```
list1=[1,2,3,4,5]
list2=list(['hello',1,2,3,'world'])
print(list1,list2,'\n')

set1={3,4,3,3,22,4,0}
set2={5,9,'yes',False,0}
print(set1,set2,'\n')

tuple1=tuple(set1)
tuple2=(33,34,44,'ok',4)
print(tuple1,tuple2,'\n')

dict1={'name':'Tom','age':2,'color':'blue'}
print(dict1)
```



二 序列

序列（列表）

序列（列表和元组，还包括字符串）的通用操作

5种操作是所有序列中通用的

1）索引

- 所谓“索引”，就是在序列中，根据所需元素的下标，返回所需元素。

```
#%%
str1='hello world'
str2=str1[2]
str3='hello world'[4]
print(str2,str3)

list1=[1,2,3,4,5]
tuple1=(33,34,44,'ok',4)
print(list1[2],tuple1[3])
```

控制台 1/A

```
In [22]: runfile('C:/Users/Administrator/Des
l o
3 ok
```

序列（列表）

序列（列表和元组，还包括字符串）的通用操作

2）切片

- 切片，就是在序列中切一块。

```
7   #%%
8   str1='hello world'
9   str2=str1[7:10]
10  str3='hello world'[1:5]
11  print(str2,str3)
12
13  list1=[1,2,3,4,5]
14  tuple1=(33,34,44,'ok',4)
15  print(list1[1:3],tuple1[1:3])
```

控制台 1/A

```
In [27]: runfile('C:/Users/Administrator/Desk
orl ello
[2, 3] (34, 44)
```

```
7   #%%
8   str1='hello world'
9   str2=str1[:10]
10  str3='hello world'[5:1]
11  print(str2,str3)
12
13  list1=[1,2,3,4,5]
14  tuple1=(33,34,44,'ok',4)
15  print(list1[1:6:2],tuple1[1:3:1])
```

控制台 1/A

```
In [33]: runfile('C:/Users/Administrator/Desk
hello worl
[2, 4] (34, 44)
```

序列（列表）

序列（列表和元组，还包括字符串）的通用操作

3）序列相加

- 就是用“+”来拼接序列。但是类型必须一致。

```
7   #%%
8   str1='hello world'
9   str2='today is the day '
10  print(str2+str1)
11
12  list1=[1,2,3,4,5]
13  tuple1=(33,34,44,'ok',4)
14  print(list1+tuple1)
15
```

控制台 1/A

```
In [41]: runfile('C:/Users/Administrator/Desktop/test/aaaa.py', wdir='C:/Users
today is the day hello world
Traceback (most recent call last):

  File "C:\Users\Administrator\Desktop\test\aaaa.py", line 14, in <module>
    print(list1+tuple1)

TypeError: can only concatenate list (not "tuple") to list
```

序列（列表）

序列（列表和元组，还包括字符串）的通用操作

4）序列相乘

- 就是用“*”来拼接序列。只能和常数乘。

```
7   #%%
8   str1='hello world'
9   str2='today is the day '
10  print(str2*5)
11
12  list1=[1,2,3,4,5]
13  tuple1=(33,34,44,'ok',4)
14  print(5*list1)
15  print(tuple1*5)
```

控制台 1/A

```
In [50]: runfile('C:/Users/Administrator/Desktop/test/aaaa.py', wdir='C:/Users/Administrator/Desktop/test')
today is the day today is the day today is the day today is the day today is the day
[1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
(33, 34, 44, 'ok', 4, 33, 34, 44, 'ok', 4, 33, 34, 44, 'ok', 4, 33, 34, 44, 'ok', 4, 33, 34, 44, 'ok', 4)
```

序列（列表）

序列（列表和元组，还包括字符串）的通用操作

5）成员资格测试

- 成员资格是指“用运算符 in 来检测指定元素是否包含于序列”
- 如果元素包含于序列，程序返回“True”；反之，返回“False”。

```
7   #%%
8   str1='hello world'
9   str2='today is the day '
10  print('h' in str1)
11  print('today' in str2)
12
13  list1=[1,2,3,4,5]
14  print('today' in list1)
15
```

控制台 1/A

```
In [55]: runfile('C:/Users/Administrat
True
True
False
```

序列（列表）

列表的专有操作

- “可修改”与“列表方法”

```
7   #%%
8   list1=[1,2,3,4,5]
9   list1[3]=10
10  print(list1)
```

控制台 1/A

```
In [58]: runfile('C:/Users/Admin
[1, 2, 3, 10, 5]
```

利用下标修改列表

```
7   #%%
8   list1=[1,2,3,4,5]
9   del list1[0:3]
10  print(list1)
```

控制台 1/A

```
In [62]: runfile('C:/Users/Adm
[4, 5]
```

利用del函数删除列表

```
7   #%%
8   list1=[1,2,3,4,5]
9   list1[0:3]=[11,12,13]
10  print(list1)
```

控制台 1/A

```
In [65]: runfile('C:/Users/Admini
[11, 12, 13, 4, 5]
```

利用切片替换

序列（列表）

列表的专有操作

- “列表方法”是指与对象（字符串，参数等）紧密联系的函数。
- 方法调用时，方法前要加上对象名和句点。

```
7   #%%
8   list1=[1,2,3,4,5]
9   list1.append(11)
10  print(list1)
11  list1.append(12)
12  print(list1)
13  list1.append(13,14)
14  print(list1)
```

控制台 1/A

```
In [73]: runfile('C:/Users/Administrator/Desktop/test/aaaa.py', wdir='C:/Use
[1, 2, 3, 4, 5, 11]
[1, 2, 3, 4, 5, 11, 12]
Traceback (most recent call last):

  File "C:\Users\Administrator\Desktop\test\aaaa.py", line 13, in <module>
    list1.append(13,14)

TypeError: append() takes exactly one argument (2 given)
```

```
7   #%%
8   list1=[1,2,3,4,5]
9   list1.append(11)
10  print(list1)
11  list1.append(12)
12  print(list1)
13  list1.append([13,14])
14  print(list1)
15  list1.extend([13,14])
16  print(list1)
```

控制台 1/A

```
In [81]: runfile('C:/Users/Administrator/D
[1, 2, 3, 4, 5, 11]
[1, 2, 3, 4, 5, 11, 12]
[1, 2, 3, 4, 5, 11, 12, [13, 14]]
[1, 2, 3, 4, 5, 11, 12, [13, 14], 13, 14]
```

append和extend

序列（列表）

列表的专有操作

- “列表方法”

```
7   #%%
8   list1=[1,2,3,4,5,1,2,3,4,5,6,7,8,1,2,3]
9   a=list1.count(1)
10  print(a)
11
12
13
```

控制台 1/A

```
In [84]: runfile('C:/Users/Administrator/Desktop/test/aa
3
```

```
6   """
7   #%%
8   list1=[1,2,3,4,5,1,2,3,4,5,6,7,8,1,2,3]
9   a=list1.index(1)
10  b=list1.index(8)
11  print(a,b)
12
13
```

控制台 1/A

```
In [87]: runfile('C:/Users/Administrator/Desktop/te
0 12
```

count和index

序列（列表）

列表的专有操作

- “列表方法”

```
7   #%%
8   list1=[1,2,3,4,5,1,2,3,4,5,6,7,8,1,2,3]
9   list2=list1
10  tuple1=tuple(list1)
11  list1.sort()
12  print(list1)
13
14  b=sorted(list2)
15  c=sorted(tuple1)
16  print(b,'\n',c)
```

控制台 1/A

```
In [94]: runfile('C:/Users/Administrator/Desktop/test/a
[1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 5, 5, 6, 7, 8]
[1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 5, 5, 6, 7, 8]
 [1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 5, 5, 6, 7, 8]
```

sort和sorted

sort无返回值，sorted是函数，有返回值，可以作用于各种序列，但只会返回列表

序列（元组）

元组的特点

- “不可更改”

```
7   #%%
8   a=1,2,3
9   b=(4,5,6)
10  print(a,b)
11
12
13
```

控制台 1/A

```
In [104]: runfile('C:/Users
(1, 2, 3) (4, 5, 6)
```

有无逗号是计算机识别是否为元组的依据，圆括号反而无关紧要

序列（元组）

元组的特点

- 列表里的方法大部分依然可用，不过需要更改的方法不可用

```
7   #%%
8   a=1,2,3,4,5,6,7,8,2,2
9   print(a[3])
10  print(a[3:6])
11  print(a.count(2))
12  print(a.index(2))
13
```

控制台 1/A

```
In [112]: runfile('C:/Users/Administ
4
(4, 5, 6)
3
1
```

```
6   """
7   #%%
8   a=1,2,3,4,5,6,7,8,2,2
9   print(a[3])
10  print(a[3:6])
11  print(a.count(2))
12  print(a.index(2))
13  print(a.append(2))
```

控制台 1/A

```
In [115]: runfile('C:/Users/Administrator/Desktop/test/aaaa.py'
4
(4, 5, 6)
3
1
Traceback (most recent call last):

  File "C:\Users\Administrator\Desktop\test\aaaa.py", line 13,
    print(a.append(2))

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

序列（元组）

元组的特点

为什么我们要用元组

1. 元组比列表操作速度快。定义了一个值，仅需要不断的遍历，需要使用元组

2. 若定义数据是常量，需要使用元组

3. 元组不可变，可以作为字典的键（key）

三　字典

字典

字典的创建

- 字典通过key和value创建，前“键”后“值”，合起来称之为“项”。
（“键”必须是独一无二的，“值”则可以重复）

字典是 Python 中的唯一内置映射

```
7   #%%
8   a=1,2,3,4,5,6,7,8,2,2
9   b=dict(ss=a,tt=[1,2,3,4,5,6,7,8,9,10,11,13])
10  print(b)
11  c={'1':'a','2':'b'}
12  print(c)
13  d={'a':[1,2,3],'b':[a,b,c]}
14  print(d)
```

控制台 1/A

```
In [126]: runfile('C:/Users/Administrator/Desktop/test/aaaa.py', wdir='C:/Users/Administrator/Desktop/test')
{'ss': (1, 2, 3, 4, 5, 6, 7, 8, 2, 2), 'tt': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13]}
{'1': 'a', '2': 'b'}
{'a': [1, 2, 3], 'b': [(1, 2, 3, 4, 5, 6, 7, 8, 2, 2), {'ss': (1, 2, 3, 4, 5, 6, 7, 8, 2, 2), 'tt': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13]},
{'1': 'a', '2': 'b'}]}
```

字典

字典的创建

- 字典的基本操作与序列类似

```
 7 a={'a':'12','b':13,'c':True}
 8 print(a)
 9 print(len(a))
10 print(a['a'])
11 a['a']=15
12 print(a)
13 del a['c']
14 print(a)
15 print('b' in a)
```

IPython console

Console 1/A

```
In [11]: runfile('C:/Users/isaac/.spyde
{'a': '12', 'b': 13, 'c': True}
3
12
{'a': 15, 'b': 13, 'c': True}
{'a': 15, 'b': 13}
True
```

字典

字典

一些重要的不同之处

- **键的类型：**与序列的索引不同，字典的键不限于整数，任何不可变的数据类型皆可以为键。
- **自动增加：**可以直接将字典不含有的项加入字典，而不需像序列一样用取代或函数的方式。
- **成员资格：**a in d 是指 a 是否存在于字典 d 的键中，而不是值中。而在序列中，in 用来查找相应的值。

字典

字典

一些重要的不同之处

- **自动增加：**可以直接将字典不含有的项加入字典，而不需像序列一样用取代或函数的方式。

通过程序运行我们可以看到，无法直接在列表中加入一个新的元素。但是对于字典，我们可以直接加入一个新的项。

```
dict1={'a':'12','b':13,'c':True}
dict1['d']=15
print(dict1)

list1=[5,6,7,8]
list1[1]=4
print(list1)
list[4]=9
```

```
IPython console
Console 1/A

In [17]: runfile('C:/Users/isaac/.spyder-py3/temp.py', wdir='C:/Use
{'a': '12', 'b': 13, 'c': True, 'd': 15}
[5, 4, 7, 8]
Traceback (most recent call last):

  File "<ipython-input-17-53a1beb2ab35>", line 1, in <module>
    runfile('C:/Users/isaac/.spyder-py3/temp.py', wdir='C:/Users/isa

  File "C:\ProgramData\Anaconda3\lib\site-packages\spyder_kernels\cu
    execfile(filename, namespace)

  File "C:\ProgramData\Anaconda3\lib\site-packages\spyder_kernels\cu
    exec(compile(f.read(), filename, 'exec'), namespace)

  File "C:/Users/isaac/.spyder-py3/temp.py", line 14, in <module>
    list[4]=9

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

字典

字典方法

```
7 dict1={'a':'12','b':13,'c':True}
8 dict1.clear()
9 print(dict1)
10
```

IPython console

Console 1/A

```
In [22]: runfile('C:/Users/isaac/.spyder
{}
```

```
7 dict1={'a':'12','b':13,'c':True}
8 dict2=dict1.copy()
9 print(dict2)
10
```

IPython console

Console 1/A

```
In [26]: runfile('C:/Users/isaac/.spyder
{'a': '12', 'b': 13, 'c': True}
```

```
6 """
7 dict1={'a':'12','b':13,'c':True}
8 key1=dict1.keys()
9 print(key1)
10
```

IPython console

Console 1/A

```
In [30]: runfile('C:/Users/isaac/.spyder-py
dict_keys(['a', 'b', 'c'])
```

```
7 dict1={'a':'12','b':13,'c':True}
8 value1=dict1.values()
9 print(value1)
10
```

IPython console

Console 1/A

```
In [34]: runfile('C:/Users/isaac/.spy
dict_values(['12', 13, True])
```

```
7 dict1={'a':'12','b':13,'c':True}
8 dict2={'a':18,'b':19,'c':20,'d':21}
9 dict1.update(dict2)
10 print(dict1)
11
```

IPython console

Console 1/A

```
In [3]: runfile('C:/Users/isaac/.spyder-py
{'a': 18, 'b': 19, 'c': 20, 'd': 21}
```

四 集合

集合

集合类似于数学概念，又像是没有value的字典

特点是无序和不能重复

- 常用方法包括add，clear，remove，intersection，difference，union

```
7 set1={'a','12','b',13,'c',True}
8 set2={'a','b','c','d'}
9 set1.add(15)
10 print(set1)
11 set2.remove('c')
12 print(set2)
```

```
IPython console
Console 1/A

In [12]: runfile('C:/Users/isaac/.spyd
{True, 'a', '12', 13, 15, 'c', 'b'}
{'a', 'd', 'b'}
```

集合

- 常用方法包括add，clear，remove，intersection，difference，union

```
 7 set1={'a','12','b',13,'c',True}
 8 set2={'a','b','c','d'}
 9
10 print(set1.intersection(set2))
11 print(set1.difference(set2))
12 print(set1.union(set2))
```

IPython console

Console 1/A

```
In [17]: runfile('C:/Users/isaac/.spyder-p
{'a', 'c', 'b'}
{True, '12', 13}
{True, 'd', '12', 'a', 13, 'c', 'b'}
```

五 总结

总结

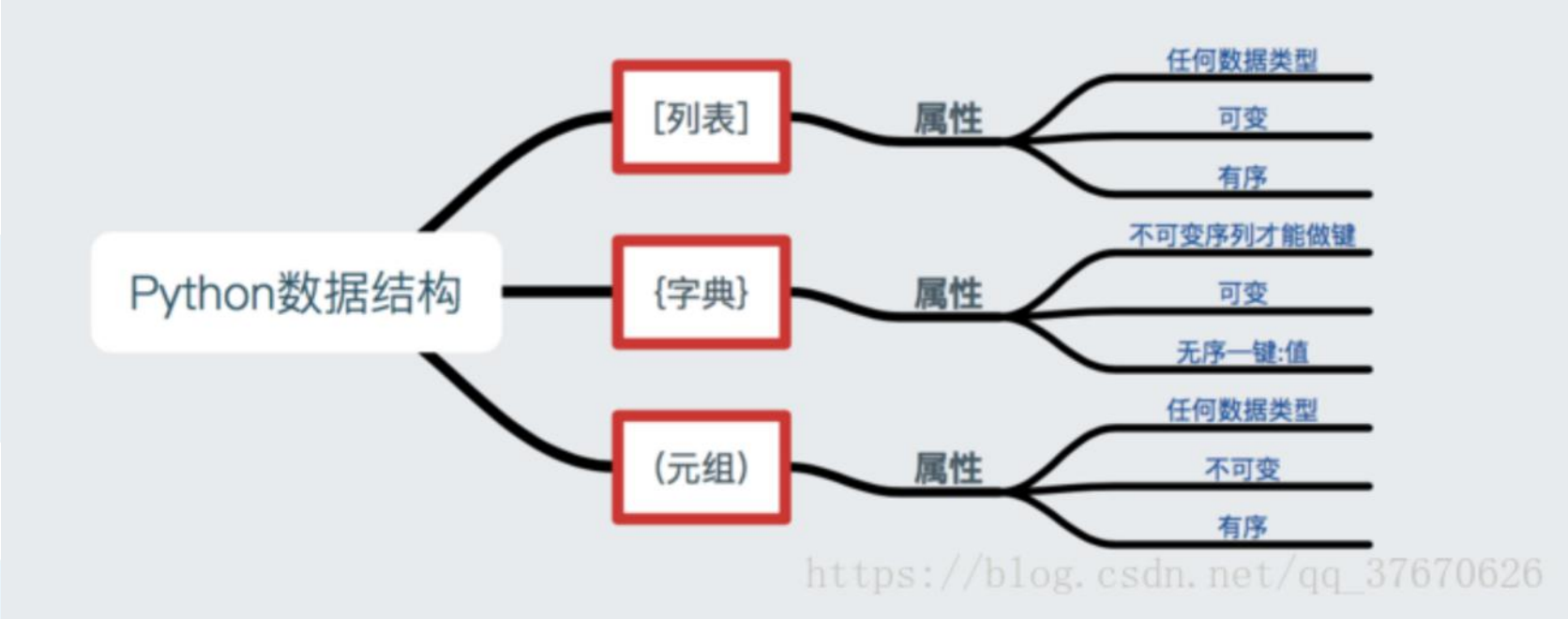
四种数据类型的比较

- 元组Tuple是存放固定的数据
- 集合set中的数据插入和遍历的时间，随数据的增多而变慢
- 列表List中的数据插入和查询的时间，随数据的增多而变慢
- 字典Dict中的数据插入和查询的速度非常快，不会因为数据太多而变慢

元组、集合和列表占用内存较少，字典占用内存较多，字典是一种通过占用空间来换取操作速度的一种数据类型。

总结

四种数据类型的比较



六 实例

实例

```python
def getNum(): #获取用户不定长度的输入
    nums = []
    iNumStr = input("请输入数字(回车退出): ")
    while iNumStr != "":
        nums.append(eval(iNumStr))
        iNumStr = input("请输入数字(回车退出): ")
    return nums

def mean(numbers): #计算平均值
    s = 0.0
    for num in numbers:
        s = s + num
    return s / len(numbers)

def dev(numbers, mean): #计算标准差
    sdev = 0.0
    for num in numbers:
        sdev = sdev + (num - mean)**2
    return pow(sdev / (len(numbers)-1), 0.5)

def median(numbers): #计算中位数
    new = sorted(numbers)
    size = len(numbers)
    if  size % 2 == 0:
        med = (new[size//2-1] + new[size//2])/2
    else:
        med = new[size//2]
    return med

n = getNum()
m = mean(n)
print("平均值:{},标准差:{:.2},中位数:{}.".format(m, dev(n,m),median(n)))
```

```
In [5]: runfile('C:/Users/Administrator/Desktop/test/

请输入数字(回车退出): 15

请输入数字(回车退出): 32

请输入数字(回车退出): 56

请输入数字(回车退出): 88

请输入数字(回车退出): 12.5

请输入数字(回车退出): 68.5

请输入数字(回车退出): 14.5

请输入数字(回车退出):
平均值:40.92857142857143,标准差:3e+01,中位数:32.
```

感谢参与 下堂课见