集合的集运算(重点)

```
集合的交集,并集,差集,对称差集的运算
{1,2,3,4}交集{4,5,6,7} -->4    交集(&)
{1,2,3,4}并集{4,5,6,7} ->1,2,3,4,5,6,7  并集(|)
{1,2,3,4}差集{4,5,6,7} ->1,2,3  差集(-)
{1,2,3,4}对称差集{4,5,6,7}->1,2,3,5,6,7 4同时出现在两个集合中,不选它 对称差集(^)

set1={1,2,3,4}
set2={4,5,6,7}
print(set1&set2)#交集
print(set1|set2)#并集
print(set1-set2)#差集
print(set1^set2)#对称差集


#案例:使用花括号和set创建各一个集合,然后对集合进行交、并、差、对称差运算



```

字典(重点)

```
字典(Dictionary)的简介
1.字典是一种数据类型
2.列表是有序的对象集合，字典是无序的对象集合。两者之间的区别在于：字典当中的元素是通过键来存取的，而不是通过偏移存取
3.字典的元素是键值对,键(key)必须使用不可变类型,同一个字典中，键(key)建议是唯一
比如:{1:"hello",2:"world"}

字典的创建
1.创建字典:d={1:"hello",2:"world"}
2.创建空字典:d2={}

#案例:从终端输入,输入格式为key:value模式,如:name:maker,然后把数据存储到字典中




```

字典的访问(重点)

```

d={1:"hello",2:'world'}
print(d)
```

```
#通过键获取值,下面的1不是下标,是键
print(d[1])#hello
#通过get函数也可以获取键对应的值
print(d.get(1))#hello
d2={'a':"hello","b":"world"}
print(d2["a"])#hello
#获取字典中所有的键
print(list(d.keys()))
#获取字典中所有的值
print(list(d.values()))

# print(d[7])#报错
print(d.get(7))#None

#案例:有字典dict={1:'悟空',2:'短笛',3:'贝吉塔',4:'小林',5:'冰河'}
#请把字典dict的键值取出,赋值给list1,把值取出,赋值给list2,再list1尾部添加66,list2尾部添加'星失'



```

字典的增删改(重点)

```
字典的添加修改删除
1.添加:
通过[]来增加键值对
d={}
d[1]="hello"
print(d)#{1: 'hello'}
d[2]="world"
print(d)#{1: 'hello', 2: 'world'}
vv={3:'aaa',4:'bbbb',5:'ccccc'}
d.update(vv)
print(d)#{1: 'hello', 2: 'world', 3: 'aaa', 4: 'bbbb', 5: 'ccccc'}
2.修改
当字典中有这个键,那么增加就变为修改
d={1:"hello",2:'world'}
d[1]="helloaaaa"
print(d)#{1: 'helloaaaa', 2: 'world'}

3.删除
#注意：删除指定的key，则对应的value也会随着被删除
print(d1.pop(3))     #返回删除的key对应的value
d1.popitem()         #随机返回并删除字典中的一对键和值(一般删除末尾对)
print(d1)
del d1[1]    #删除元素
print(d1)
#del d1 #删除字典

#d1.clear        #清空字典

```

```
d={1:'a',2:'b',3:'c'}
print(d.pop(1))#a
print(d)#{2: 'b', 3: 'c'}

print(d.popitem())#(3, 'c')
print(d)#{2: 'b'}

del d[2]
print(d)#{}

d={'a':"hello",'b':"world"}
print(d)#{'a': 'hello', 'b': 'world'}
d.clear()
print(d)#{}
del d
print(d)#报错

#案例:把1-100的单数当成字典的键,偶数当成字典的值,创建一个字典,然后增加{'name':'maker'},
#{1:'one'}数据
#案例:上一题的基础上,删除11键元素,返回他的值,然后打印.随机删除上一题的一对键值对,并打印
#然后清空并删除字典


```

字典的遍历(重点)

```
dict1 = {"name":"jbb","sex":"man","age":18}

dict.copy() 返回一个字典的深复制
d1={1:"a",2:'b',3:'c',4:'d'}
d2=d1.copy()
print(d2)
d3=d1
print(d3)
print(id(d1))#31358408
print(id(d2))#31358480
print(id(d3))#31358408

d3[1]="aaa"
print(d1)#{1: 'aaa', 2: 'b', 3: 'c', 4: 'd'}
print(d2)#{1: 'a', 2: 'b', 3: 'c', 4: 'd'}

dict.fromkeys(seq, value) 创建一个新字典，以序列seq中元素做字典的键，val为字典所有键对应的初始值
dict1.get(key, None) 返回指定键的值，如果值不在字典中返回default值
dict1.keys() 以列表返回一个字典所有的键
dict1.values() 以列表返回字典中的所有值
dict1.items() 以列表返回可遍历的(键, 值) 元组数组
#把字典转换为列表,列表中的元素是元组形式的键值对


```

```
print(list(dk.items()))#[(1, 100), (2, 100), (3, 100), (4, 100), (5, 100), (6, 100), (7, 100), (8, 100), (9, 100)]

案例:
1、逐一显示指定字典中的所有键，并在显示结束之后输出总键数
2、   list1 = [1,2,3,4,5,6,7],
    list2 = ["星期一","星期二","星期三","星期四","星期五","星期六","星期日"]
    以list1中的元素作为key，以list2中的元素作为value生成一个新的字典dict2。
```

数据类型总结

```
数字型
int、float
特点:
1.除法时,总是返回浮点数
2.可以使用//来取整


字符串
1.单引号或双引号包含的都叫字符串
2.单包双,双包单
3.字符串中的字符不能改变

列表
1.列表是用[]包含的数据
2.列表里的元素可以是不同数据类型的
3.可以通过下标来获取或添加数据,但不能越位
4.追加元素(append),添加序列(extend),插入数据(insert),删除(pop,remove),清空(clear),倒序(reverse),排序(sort),长度(len),最大(max),最小(min),获取下标(index),出现次数(count)


元组
1.元组是用()包含的数据
2.元组里的元素可以是不同数据类型的
3.元组里的元素不可以改变
4.定义一个元素的元组:tup(元素,)
5.可以删除元组,但不能删除元组里的元素
6.操作api,长度(len),最大(max),最小(min),获取下标(index),出现次数(count)
7.当元组赋值给若干个变量时,变量的个数没有元组里元素个数多的话,带*的变量,会接收剩下的数据,这时,带*变量就是列表


集合
1.集合是用{}包含的数据
2.集合是个无效且不重复的,不能通过下标来访问
3.作用:清除重复数据,或判断成员
```

```
4.操作api:添加(add,update),删除(remove,pop,discard),删除集合(del),长度(len)
5.集运算:交集(&),并集(|),差集(-),对称差集(^)


字典
1.字典是用{}包含的数据,但数据是个键值对
2.字典是个无效的,也就是说不能通过下标来访问
3.访问:字典[键]->值
4.获取所有键(keys),获取所有值(values)
5.字典的操作:增加([],updae),删(pop,popitem,del),清空(clear),删除字典(del),拷贝(copy),
序列为键,某个数为值(fromkeys),返回列表,元素是元组(items)


```

函数的定义(重点)

```
1.什么是函数
函数是组织好的,可重复使用的,用来实现单一或相关功能的代码块

2.函数有内建函数和用户自定义函数
3.定义函数的格式:
def 函数名():
    代码块
调用函数:
    函数名()

#定义函数的时候写的参数叫形参
def mytest(name):#name="maker"
    print(name)

#调用函数时写的参数叫实参
mytest("maker")


def mytest02():
    name="maker"
    return name#把name里面存储的数据返回到函数外面

myname=mytest02()
print(myname)
#步骤:
'''
1.代码运行到myname=mytest02(),先执行等号的右边
2.执行mytest02函数,当函数执行完成后,myname=mytest02()中的mytest02()就变为函数的返回值
3.myname="maker"
4.print(myname)
'''

案例:
1.声明一个函数，实现求1+2+3+...+N的和,如:终端输入10,函数返回1-10的和
```

```
def mysum():
    n=int(input("请输入数字:"))
    sum=0
    for i in range(1,n+1):
        sum+=i
    print("总和为:",sum)

mysum()

2.写一个函数add，接受两个整数作为参数，返回这两个整数的和。
def add(a,b):#a=10,b=20
    n=a+b
    return n

m=add(10,20)
print(m)#30


```

参数类型(重点)

```
参数类型有,必需参数,关键字参数,默认参数,不定长参数

1.必需参数
形参有多少个,实参必须有多少个
def add(a,b):
    return a+b

# add(10)#报错
add(10,20)


#案例:定义一个函数，有三个参数，从终端输入3个数，传入到这个函数，并调用这个函数，打印出参数的内容

2.关键字参数
使用关键字参数允许函数调用的时候实参的顺序和形参的顺序可以不一致，可以使用关键字进行自动的匹配
def Mymsg(name,age,sex,height):
    print("name=",name)
    print("age=",age)
    print("sex=",sex)
    print("height=",height)
#如果调用时,使用了关键字参数,那么后面的参数都要用关键字参数
Mymsg("maker",sex="男",age=18,height="180cm")

#案例:定一个函数，参数有id,name,age,ipthon,有一个字典
#{'id':1,'name':'maker','age':18,'ipthon':'123234324423'},获取对应的值，传入函数中

3.默认参数
如果形参有值,那么实参可传可不传,不传,默认使用形参的值,如果传,那么使用实参的值
如果形参有默认值,那么这个形参后面的参数都必须有默认值
def myadd(a,b=20):
    return a+b
```

```

print(myadd(20))#40
print(myadd(20,50))#70

#案例:定一个函数,打印用户的信息，用户的信息有姓名，性别，年龄，电话，其中性别默认为'男'

4.不定长参数
你可能需要一个函数能处理比当初声明时更多的参数。这些参数叫做不定长参数。
把参数打包成元组给函数调用,如果在函数调用时没有指定参数，它就是一个空元组
def mytest(*arr):
    print(arr)#(1,2,3,4,5)

mytest(1,2,3,4,5)


def mytest02(**arr):
    print(arr)##{'name': 'maker', 'age': 18}

mytest02(name="maker",age=18)


#案例:实现一个函数，支持传入任意多个整数进行加法运算，并返回结果
def myadd(*arr):
    sum=0
    for i in arr:
        sum+=i
    return sum

print(myadd(1,2,3,4,5))


```

参数传递方式(重点)

```
1.不可变类型：如 整数、字符串、元组。如fun（a），传递的只是a的值，没有影响a对象本身
2.可变类型：如 列表，字典。如 fun（la），则是将 la 真正的传过去，修改后fun函数外部的la也会受影响

1.实参传人函数中,实参不会改变
#形参的a和实参的a不是同一个
def mytest(a):#a=10
    a=20

a=10
mytest(a)#mytest(10)
print(a)

2.实参传入函数中,实参会改变
def mytest(mylist):#接收的也是空间地址
    print(id(mylist))
    mylist.append(100)

mylist=[1,2,3]
```

```
print(id(mylist))
mytest(mylist)#传递的是空间地址
print(mylist)

#案例:定义一个字典,键是1,2,3,4,5,值都是0,通过一个函数改变这个字典,让他的值为10,20,30,40,50

```

return语句(重点)

```
作用：表示一个函数执行完毕之后得到的结果返回给调用者
return后面没有什么的语句返回None
格式:
def 函数名():
    函数体
    return 表达式或值或多个值或没有什么

def mytest():
    # return 10
    # a=10
    # return a
    # a=10
    # b=20
    # return a+b
    return 1,2,3,4#元组
    #return是表示函数结束,return后面的代码就不会执行

print(mytest())

#案例:定义一个函数,这个函数返回多个值

```

作用域

```
作用域:就是你定义的变量有效果的范围
变量有全局变量,有局部变量
#全局变量,从定义开始到文件结尾都有效果
a=10

def mytest():
    print(a)

for i in range(a):
    print(i)
    print(a)

if a>5:
    print(a)

mytest()
```

```
17
18
19  #局部变量,在函数内定义的变量叫局部变量
20  def mytest():
21      a=10#从定义开始到函数结束有效果
22      print(a)
23
24  mytest()
25  # print(a)#报错
26
27
28
29  #在for循环,if语句,while里的定义的是全局变量
30  for i in range(10):
31      b=20
32
33  print(b)
34
35  if True:
36      c=30
37
38  print(c)
39
40  while c<31:
41      d=30
42      c+=1
43  print(d)
44
45  当全局变量和局部变量同名时
46  a=10
47
48  def mytest():
49      # print(a)#报错
50      # a=20
51      # print(a)#20
52      #如果想要在函数内修改全局变量,那么可以使用global来声明
53      global a
54      a=30
55
56      #如果要使用全局的a,那么可以使用传参
57
58
59  print(a)#10
60  mytest()
61  print(a)
```