**1.列表**

前面我们了解了str、int、float等数据类型，比如人名用str，name='大雷'，那如果我们一个部门呢？还用字符串存数据么，names='大雷,云云,星星'这样存数据，在调用的时候岂不是很麻烦，比如我们想取到星星，还要用到字符串方法，会比较麻烦，或者你想每个人取一个字符串的变量名，这样也麻烦啊，有多少个人，岂不是就要顶一个多少个变量了？为了解决这些麻烦，就有了列表，顾名思义，就是一个表，里面存放一系列数据，python里的列表定义比其他语言要简单，中括弧[]括起来就是列表了，当然在python里列表也叫数组，大家不要因为名字不同而困惑，都一个概念，列表增删改查如下：

1 #列表

2 names=['小刚','小红','小明','小小','猛犸'] #列表命名，列表名=[元素1,元素2,元素3]，元素用英文逗号隔开，元素如果是字符串需要加引号

3 #0 #1 #2 #3 #4

4 #列表里的元素是有排序的，所以每个元素都有一个数字编号，计算机是从0开始的，所以['小刚','小红','小明','小小','猛犸']这些元素对应的索引依次为：01234,当然在python里除了索引这个叫法，还有下标、角标，都一个意思

5

6 #列表取值

7 print(names[0]) #列表名[元素索引]，这个格式就取到索引对应的元素了，这里取到'小刚'，列表只有这一种取值方式

8 print(names[-1]) #负数表示从后往前取元素，这里取到了猛犸

9

10 #新增元素

11 names.append('云云') #列表名.append(元素)，这个方法是在列表最后面加一个元素，这里在列表最后加上'云云'

12 names.insert(2,'李白') #列表名.insert(索引值,元素)，这个方法是在指定索引位置插入元素，之前该位置及后面的额元素向后移一位，这里是在列表的第三个位置插入'李白'

13

14 #修改元素

15 names[0]='大刚' #列表名[元素索引]=新元素，这样就修改元素了

16

17 #删除元素

18 names.pop() #默认删除最后一个元素

19 names.pop(2) #删除指定下标的元素

20 names.remove('小红') #和pop不同，删除列表中指定的元素

21 del names[2] #删除指定下标的元素

22 names.clear() #清空列表

23

24 #多维数组

25 names1=['小刚','小红',['宙斯','火女'],'小明','小小','猛犸'] #数组里再来一个数组就是二维数组了

26 names2=['小刚','小红',['宙斯',['女王','拉比克'],'火女'],'小明','小小','猛犸'] #组二维数组里再来一个数组就是三维数组了，依次类推

27 print(names2[2][1][1]) #数组里面的数组就当元素来取，取出来之后，又按数组取值方式取下去就好了，这里取到了'拉比克'

28

29 print(names)

　　列表内置方法：

1 #内置方法：

2 names=['小刚','小红','小明','小小','猛犸']

3 names2=['宙斯','火女']

4 names.extend(names2) #合并两个list。将names2合并到names里面

5 names+names2 #合并两个list。但和上面不一样，这里只是把两个列表连接起来，原来的列表不会变化，类似字符串相加

6 new\_name=names+names2 #用+的时候，需要用一个新列表来接收

7 print(names.count('小小')) #统计元素出现的次数

8 print(names.index('小小')) #查询到元素的索引

9 names.reverse() #反转数组

10 names.sort() #数组排序，升序

11 names.sort(reverse=True) #数组排序，降序

12 print(names)

　　列表切片，顾名思义，就是把列表切成我们想要的片段。上面了解到，列表是有下标、索引的，那么切片就是根据这个下标来进行切片的，比如我想去第一个元素到第三个元素，那么这样写：names[0:3]，这样就取到前面三个元素了，但是你会发现，3的下标是第四个元素，为啥切片要切到3呢，因为python切片里，顾头不顾尾，意思就是从开始下标取到结束下标的前一位元素。另外如果是从0下标开始取值，那么可以省略不写，直接写[:3]即可；如果是取值到最后，那么可以省略不写最后的索引，直接写[3:]即可，显然[:]这样就是取所有的了。还有一个步长的概念，就是每走几步取一个元素，比如列表num=[1,2,3,4,5,6,7,8,9]，num[0:10:2]，这里就是每两个元素取一个，取的结果是：[1,3,5,7,9]。代码如下：

1 #列表切片

2 nums=[1,2,3,4,5,6,7,8,9]

3 print(nums[1:3]) #取第二个元素到第三个元素，也就是2,3

4 print(nums[3:]) #取所有的元素，一直取到最后，后面的下标可以省略不写

5 print(nums[:6]) #取所有的元素，从头开始取，前面的下标可以不写

6 print(nums[0:10:2]) #步长2，每两个元素取一个，这里取到1,3,5,7,9

7 print(nums[::1]) #取所有的元素,1就是没个元素都取

8 print(nums[::-1]) #步长为负数的情况下是从后往前取

9 print(nums[7:1:-2]) #步长为负数时，这里的下标必须是前面的下标大于后面的下标，因为他是从后往前取的

**2.元祖**

　　元祖，跟列表的定义是一样的，只是元祖是用小括弧()，列表用的是中括弧[]，在命名上就这一点区别，那为什么有了列表，还来一个元祖，列表就够用了啊，反正都差不多，这里是因为元祖比列表更特殊一些，元祖命名后就不能增、删、改，就已经写死了，不能动他，你只能从里面取元素，只有读取的权限。有什么用呢？像我们在开发的时候，有一些基础数据，比如数据库连接信息，这些事定死的，你改了就连接不上了，所以这里用到元祖，避免后面调试啥的误操作，增删改了数据库连接信息，导致连接失败。元祖只有三个方法(其他方法都不适用)：

#元祖

num=(1,2,3,4,5,6,6)

print(num[0]) #通过下标取值

print(num.count(6)) #统计元素出现的次数

print(num.index(6)) #查询到元素的索引

　　既然说到了元祖，这里了解下可变变量和不可变变量，可变变量就是命名之后，还可以对他的内容进行增、删、改等操作，不可变变量就是，命名之后就不可以增、删、改他的内容了，像这里的元祖就是不可变变量，一旦命名就不可以对他的内容进行增、删、改等操作，只有读取的操作权限，所以元祖只有上面的三个方法，列表里的增删改都不适用于元祖，用的话会报错。类似于元祖这样的不可变变量还有字符串，字符串的内容是不可以增、删、改的，只可以读取。

**3.字典**

　　字典的概念大家再熟悉不过了，上小学就用到字典了，每天书包里都揣已字典，遇到不懂的就查一下不懂的这个关键字，然后后面跟很多解释，举例啥的。就像这个字典的需求，用列表可以实现吗？可以，但是很麻烦，比如这样存：

words=['菜鸟','奋斗','努力'] #中文

comments=['newbie','fight','effot'] #翻译

　　如上述代码，但刚遇到这样的需求时，我们首先要找到中文words里的的下标，然后根据下标再去comments里找对应的英文，这样是可以满足字典的需求，但是词典那么多，这样找效率特别低，因为list在找元素的是后是从前面一直找到后面，当有几万个词的时候，计算机要找很久才找的到，而且，这样存起来也不直观，看小学时候字典里那种存的多直观，前面是词，后面跟解释，所以python仿照字典弄了一个字典的样式，然后你在找词的时候，他直接找这个词，不会从前往后一个个的找，直接就找到了，他是无需的，很大程度上提高了效率。

　　字典的特性：

　　1.字典是无序的，因为它没有下标，用key来当索引，所以是无序的；

　　2.字典的key必须是唯一的，因为它是通过key来进行索引的，所以key不能重复，所以字典里不会产生重复数据。

　　字典命名也很直观，字典名={key1:value1,key2:value2}，用大括号{},里面的格式是key:value，然后用逗号隔开各个元素，这样就直接定义好了一个字典并且赋值，代码如下：

1 #字典

2 informations={'星星':['江西',18,'男',180],'大雷':['合肥',16,'男',190]} #定义字典

3 #

4 #新增

5 informations['人马']=['近卫',128,'未知',156] #新增，直接指定key=value就新增了

6 informations.setdefault('宙斯',['天灾',1280,'未知',152]) #这里的新增，如果这个key存在的话，就不修改他，不存在的话，就添加

7

8 #修改

9 informations['星星']=['湖南',22,'未知',150] #修改，直接指定key=value就修改了，就是说如果字典存在这个key，就修改value，不存在就新增，因为字典是不能重复key的

10 informations.update(大雷=['天灾',1280,'未知',152]) #update也可以修改，但是注意，这里的key如果是字符串的话不需要加引号

11

12 #删除

13 del informations['星星'] #删除，和list一样，list是删除指定下标的元素，这里是删除指定key的value

14 informations.pop('人马') #删除，和list一样，list是删除指定下标的元素，这里是删除指定key的value

15 informations.popitem() #随机删除一个值

16

17 #查询，直接取和通过get取的区别是，当key不存在时，get方法取值不会报错，返回None；而直接取会报错，就这一个区别

18 print(informations['人马']) #通过key取value

19 print(informations.get('人马')) #用字典的get方法通过key取value

20

21 #字典内置方法

22 print(informations.keys()) #获取字典的所有key

23 print(informations.values()) #获取字典的所有value

24 print(list(informations.keys())) #强制类型转换为列表

25 print(list(informations.values())) #强制类型转换为列表

26 print(informations.items()) #将字典转换成一个列表，一般不这样用，字典本来就快鱼list，还转换成list，没必要

27 additional={1:'a','大雷': ['火星', 16, '男', 190]}

28 informations.update(additional) #update方式是更新字典，没有的key直接新增，有可以key的话，更新value

29 print(informations)

　　字典嵌套，这个也很常用，就拿上述例子来说，首先是人名，然后是这个人的一些描述，描述内容少还好，但是，内容多的话呢，比如有身高、体重、爱好等等，太多了，用list存是不是又存在之前的问题，效率低，而且不方便取，每次取的时候还要要确定我要取得这个属性的下标然后才取得到。这里就引进了字典嵌套，用法也比较简单，如下：

informations={

'星星':{

'地址':'江西',

'age':18,

'sex':'男',

'heights':180

},

'大雷':{

'地址':'合肥',

'age':22,

'sex':'未知',

'heights':190}

}

　　这样就直观很多了，不管你有多少属性，这样加下去就行了，取值方法参考字典的取值和多维数组的取值，这里不赘述了。

**4.字符串方法**

这里没什么技巧，记住这些方法就可以了，以后要用到，常用的有下面几种方法(一定要记住有这些方法，具体怎么用可以再查)：

1 # 字符串方法

2 file\_name='D:\1自动化\day2\day2笔记.txt'

3 name='xiaoagng'

4 print(file\_name.endswith('.png')) # 判断是否以x结尾

5 name.format(name='niuniu', age=18) # 这个是格式字符串

6 print('122'.isdigit()) # 是否是数字

7 print(''.join(['hehe', 'haha', 'ee'])) # 拼接字符串

8 # 1、把list转成字符串

9 # 2、把list里面每个元素连起来

10 print('adbefF'.lower()) # 变成小写

11 print('adbefF'.upper()) # 变成大写

12 print('\nmysql \n'.strip()) #

13 print('mysql is db.'.replace('mysql', 'oracle'))

14 print('1+2+3+4'.split('+')) # 切割字符串，返回一个list

15 print('1+2+3\n1+2+3+4'.splitlines()) # 按照换行符分割，返回一个list