1. 列表

1.1. 列表的基本概述

列表是一个可变的有序的数据集合,对于列表的数据类型的特点:

① 列表的创建方式 a = [1,3,5,6,9]

② 列表是有序的集合,可以通过索引,分片进行元素的操作

③ 列表是一个可变对象,对于列表中的每个元素,可以进行修改,删除等操作

④ 列表的中元素的数据类型可以是任意的数据类型,数字,字符串

⑤ 列表可以嵌套,列表中的元素可以是列表数据类型

⑥ 列表中存放的是每个元素的对象的引用

⑦ 列表中的数据的表示方式是使用一对中括号[],每个元素之间使用逗号隔开

```
list1 = [] #空列表集合
list2 = [1,2,3,4,5] # 含有 5 个元素的列表集合
list3 = [1,3,"a","bc"] # 列表中的每个元素都可以是任何类型的数据
list4 = [1,2,[1,3,5],[2,"a","b"]] # 列表可以嵌套
```

1.2. 索引和分片操作

和字符串数据类型一样,对于列表也是可以通过索引进行列表元素的操作,也可以进行对应的切片操作,但是和字符串不一样的地方是对于字符串是不能修改里面的字符,但是对于列表是可以修改列表中的字符

```
# 列表
list1 = [1,2,4,[5,6],"a"]
# 通过索引访问第三个元素 从左往右 从 0 开始
print(list1[2])
# 通过索引访问倒数第二个元素 从右往左 从-1 开始
print(list1[-2])
# 修改 list1 中的元素 通过索引
list1[2] = 100
print("修改后的 list1",list1)
# 删除 list1 中的第二个元素
del list1[2]
print("删除后的 list1",list1)
# 打印元素个数(list)的长度
print(len(list1))
list1 = [1,2,4,[5,6],"a"]
# 切片操作 返回一个新的列表
list2 = list1[0:3]
print(list2)
# 切片修改 替换操作
# list1[0:3] =[0]
```

```
# list1[0:3] = [100,200,300]
# 删除操作 插入的是一个空的列表中的元素
# list1[0:3] = []
# 插入操作 设置开始索引和结束索引一直,然后在开始索引的位置上插入新的值
list1[2:2] = [300,400,500]
# 在最后插入一个元素,如果开始索引超过结束索引位置,那么在列表末尾添加元素
list1[len(list1)+100:len(list1)]=[9000]
print(list1)
# for 循环变量
for item in list1 :
    print(item)
```

1.3. 列表的操作符(+,*)

```
"""
+ 序列连接符
* 赋值符号
"""
# 把一个序列对象转换为列表对象
print(list("abc"))
# 报错 TypeError: can only concatenate list (not "str") to list
# print([1,2,3]+"abc")
# 正确方式 可迭代的数据类型 字符串, 列表
print([1,2,3]+list("abc"))
list1 = [1]
list2 = [[4]] * 4
print(list2)
# list2[0] = 100 的区别
list2[0][0]=100
print(list2)
```

1.4. 常用方法

```
"""
常用方法
x.append(y) 往序列 x 中添加一个元素 y 改变 x 的结果,返回 None
x.extend(y) 把 y 序列中的每个元素分别添加到 x 需要中 改变 x 的结果,返回 None
x.insert(index,y) 在索引 index 位置处插入 y 元素 index 值错误,在开头或者末尾插入元素
x.index(y) 查找 y 元素在 x 中的索引位置,如果没有找到,报错
x.count(y) 返回 y 元素在 x 中出现的次数
x.sort() 对序列 x 进行排序
x.reverse() 对序列 x 进行反转
x.remove(y) 从列表 x 中删除 y 元素
"""
list1 = [1,2,3,4]
list1.append(100)
```

```
list1.extend([10,20,30])
list1.insert(0,300)
print("100 的位置:",list1.index(100))
print("30 的出现次数:",list1.count(30))
list1.sort()
list1.reverse()
list1.remove(300)
print(list1)
```

1.5. 列表推导式

```
list1 = [1,2,3,4,5]
# list2 =list(list1)
# list2 = list1[:]
# list2 = []
# for item in list1:
#     if item > 3:
#       list2.append(item)
# 列表推导式
list2 = [item+100 for item in list1 if item > 3]
print(list2)
#
#sex = "MF"
#size = "SMLX"
#codes = []
#for i in sex:
#    for j in size:
#        if i ="F" and j == "X":
#            continue
#        codes.append(i+j)
codes = [sex+size for sex in "MF"
                  for size in "SMLX"
                      if not(sex =="F" and size == "X")]
print(codes)
```

2. 元组

2.1. 元组的基本概述

元组是一个不可变的序列,其中操作方式和字符串类似,可列表相比较,主要是元组中的数据不能进行删除,修改,增加等操作

① 元组的创建方式 tuple01 = (1,2,3,4)

② 元组是一个有序的序列,可以通过索引获取元素,以及分片操作

③ 元组中的元素可以是任意类型的数据,字符串,数字,列表,元组

④ 元组可以嵌套

⑤ 元组中存放的是每个元素的对象的引用

⑥ 元组的表示使用一对(),每个元素之间使用逗号隔开

```
#空元组类型
tuple01 = ()
# 可以放入任何数据类型
tuple02 = (1,(1,2),["abs"])
# 把列表转换为函数
tuple03 = tuple([1,2,3])
# 使用元组描述一个学生的 学号和成绩
tuple04 = ("0001",100)
```

2.2. 元组的常用操作

```
tuple01 = (1,2,3)
# 对于元组中只有一个元素集合的时候,需要使用一个逗号标记为元组
# tuple02 = ("a",)
# print(tuple02)
# print(type(tuple02))
# 元组的 + 和*
# tuple02 = tuple01 + (4,5,6)
# print(tuple02)
tuple02 = tuple01 * 3
print(tuple02)
# 索引操作
# print(tuple01.index(3))
# print(tuple01.count(3))
print(tuple01[0])
# 报错
# tuple01[0] = 100
tuple03 = ([1,2],[3,4])
print(tuple03)
# 可以修改列表中的元素的值
tuple03[0][1]=100
print(tuple03)
print("*"*50)
# 分片操作
print(tuple01[-2:])
print("tuple01:",tuple01)
print(type(tuple01))
# 循环遍历
for item in tuple01:
    print(item)
```

2.3. 元组的拆包和装包

对于元组的表示,我们以后都使用小括号和逗号,但是在某些情况,其中的括号是可以省略的

把元组中的小括号省略的操作,我们称之为拆包

```
# 在赋值符号的右边,如果有多个值使用逗号该开,默认为打包为一个元组类型的数据
a = (3,4)
print(a)
# 元组的拆包,在拆包的时候,元组中的个数需要和变量的个数保持一致
x,y = (3,4)
# 报错
x,y = (3,4,5)
print(x,y)
```

3. range

range 是一个不可变的序列对象,主要是用于生产整数的序列,通常是配合 for 循环用于生成循环次数

可以把 range 对象的数据转换为 list 数据

range:只是保存了生成数据的规则,所以在数据量大的时候占用的内存空间比 list 列表小的多,推荐在大数据量的时候使用 range

```
"""
range 常见的操作

"""
# 创建一个可以生成 0-9 的数字,用于循环
for item in range(10):
    print(item)
# 创建一个生成 5-10 的数字,用于循环
print("*"*50)
for item in range(5,11):
    print(item)
# 生成 1-10 之间的所有的奇数
print("*"*50)
for item in range(1,11,2):
    print(item)
print("*"*50)
r1 = range(0,10,1)
# 获取长度
print(len(r1))
# 判断一个元素是否在 range 中
print(9 in r1)
# 打印第二个元素
print(r1[5])
```

```
# 把 range 转换为 list 集合
print("range:",r1)
print("list:",list(r1))
```

4. 集合

4.1. 集合的基本概述

对于列表保存的数据是有序的可变数据,如果在对于数据不要求顺序而且不允许重复数据的话,可以使用集合 set 对象数据类型

特点:

① 没有顺序,不能通过索引和分片操作

② 元素不允许重复

③ 可以和其他序列 list 列表,tuple 元组进行相互转换

④ 创建方式使用 set01 = {1,4,2,6,8} 使用花括号括起来,每个元素之间使用逗号隔开

⑤ 需要注意的是不能使用 set01 = {} 创建一个空集 set 集合对象数据

```
"""
集合创建
"""
# 创建集合,元素可以是任何类型
set01 = {"a",2,4,1}
# 打印的数据不一定和存放的数据一致,对于数据是没有顺序的
print(set01)
# 集合不能嵌套(只能存放可哈希的数据类型),对于可变的列表,集合都不能存放
set02 = {1,3,4,100,(12,45)}
print(set02)
set03 = {}
# 不是集合类型
print(type(set03))
```

4.2. 集合的常用操作

```
# 创建一个空集合
s = set()
# 添加元素
s.add(1)
s.add(10)
s.add("100")
#报错,不能添加列表,和集合对象
# s.add([100])
# 删除指定的元素
```

```
s.remove(10)
# 随机删除一个元素
s.pop()
# 清空集合
s.clear()
s1 = {1,2,3,4,6}
s2 = {1,4,7,8,9}
print("s1",s1)
print("s2",s2)
# 获取两个集合的交集
print("交集",s1 & s2)
# 获取两个集合的并集
print("并集",s1| s2)
# 获取两个集合的差集
print("差集",s1 - s2)
# 返回两个集合的数据,但是不能同时在 s1 和 s2 集合中的数据
print("亦或",s1^s2)
# 判断一个数据是否在集合中
print(20 in s1)
print(s)
# 利用集合去重列表中的元素
list01 = [1,2,1,4,5,3,4,2]
print("原始",list01)
s1 = set(list01)
list01 = list(s1)
print("处理后",list01)
```

4.3. 集合推导式

```
"""
"""
# set 集合的循环
s1 = {1,3,4,5}
for item in s1:
    print(item)
# 集合推导式
s2 = {item + 10 for item in s1}
print(s2)
```

5. 字典

5.1. 字典的基本概述

对于列表来说是一个有序的可变的集合,我们可以通过对应的索引去获取列表中的每个元素,但是如果使用字典的话,其中的数据是一个无序的可变集合,我们不能通过索引获取对应的元素,我们使用的是键值对

的方式保存的数据,需要通过键去获取对应的值

特点:

① 创建方式,a ={“number”:”001”,”score”:90}使用花括号表示,每个数据项都是由键值对组成,数据项和数据项之间使用逗号隔开,键值对之间使用冒号隔开

② 数据是无序的,不能通过索引,分片操作,获取字典元素需要通过键来获取,键值对是一一对应的

③ 字典是一个可变的集合,可以对字典进行添加,删除,修改等操作

④ 字典的 key 通常都是字符串类型,也可以是数字,元组等不可变类型,不能是集合 set 和列表 list 以及字典,但是对于字典的值可以是任意类型的数据

```
# 创建一个空的字典
d1 = {}
# 创建一个有两个数据项的字典
d2 = {"name":"hesj","score":98}
# 通过 dict 方法构造一个字典 一个元组转换为一个数据项
d3 =dict([('name','wolfcode'),('addr','广州')])
print(d3)
# 长度
print(len(d2))
print(d2)
```

5.2. 字典的常用操作

```
d1 = {"姓名":"张三","数学":100,"英语":90,"语文":90}
d2 = {"姓名":"李四","数学":60,"英语":70,"语文":60}
# 获取张三同学的数学成绩
print(d1["数学"])
# 把李四同学的语句成绩改为 80
d2["语文"] = 80
# 给张三同学添加一个总成绩,如果存在对应的 key,则会覆盖,否则就会新建一个数据
d1["总成绩"]=d1["数学"]+d1["英语"]+d1["语文"]
d2["总成绩"]=d2["数学"]+d2["英语"]+d2["语文"]
# 判断 d1 是否存在英语成绩, 判断是否包含需要的键(key)
print("英语" in d1)
# 删除 d1 的数学成绩
# del d1["数学"]
# 常用方法
#获取
print(d1.get("姓名"))
# 获取历史成绩,如果没有默认为 100
print(d1.get("历史",100))
# 删除总成绩并且返回
print(d1.pop("总成绩"))
# 修改成绩
```

```
d1.update([("姓名 1","jack")])
# 获取所有的 keys
print(d1.keys())
print(d2.values())
print(d2.items())
# 使用列表存放多个学生信息
list1 =[d1,d2]
for item in list1:
    print(item)
```

5.3. 字典推导式

```
d1 = {"姓名":"张三","数学":100,"英语":90,"语文":90}
d2 = {"姓名":"李四","数学":60,"英语":70,"语文":60}
# 循环遍历打印
# 获取到所有的 keys 在循环遍历
items = d1.keys()
for item in items:
    print("%s-->%s"%(item,d1[item]))
# 使用 for in 循环默认就是对所有的 keys 进行遍历
for item in d1:
    print("%s-->%s"%(item,d1[item]))
# 字典推导式
d3 ={item:d1[item] for item in d1 }
print(d3)
```

6. 图书管理系统

```
"""
图书管理系统
图书的信息 书名,数量
主要功能
① 根据书名查找图书
② 借阅图书
③ 归还图书
④ 显示所有图书
"""
# 一个图书信息使用一个字典来表示 {"书名":"Python 实战","数量":10}
# 初始化数据
books =[
    {"书名":"Python 实战","数量":10},
    {"书名":"Java 实战","数量":10},
    {"书名":"Redis 实战","数量":10}
]
print("*"*50)
print("图书管理系统")
```

```
print("1 查找图书")
print("2 借阅图书")
print("3 归还图书")
print("4 显示所有图书")
print("5 退出系统")
print("*"*50)
while True:
    opt = int(input("请输入你的选择序号"))
    if opt == 1:
        item = input("请输入你查找的图书名:")
        for book in books:
            if book["书名"] == item:
                print("您要找的图书信息:",book)
                flag = True
                break
            # 当在循环中使用了 break 跳出循环的时候,不会执行该操作
        else:
            print("没有你要找的书籍:%s"%item)
    elif opt == 2:
        item = input("请输入你借阅的图书名:")
        for book in books:
            if book["书名"] == item and book["数量"] > 0:
                book["数量"] -= 1
                print("借阅成功")
                flag = True
                break
        else:
            print("没有你要借阅的书籍:%s"%item)
    elif opt == 3:
        item = input("请输入你归还的图书名:")
        for book in books:
            if book["书名"] == item:
                book["数量"] += 1
                print("归还成功")
                break
        else:
            books.append({"书名":item,"数量":1})
    elif opt == 4:
        print("="*50)
        for book in books:
            print(book)
        print("="*50)
    elif opt == 5:
        break
```

```
    else:
        print("您输入的有误,请重新输入")
```

7. 数据类型回顾

7.1. 对象的使用

① 列表,字典,元组,集合中存放的可以是任何数据类型的对象

② 列表,字典,元组,集合中可以任意的嵌套

③ 列表,字典,集合可以动态的扩大和缩小

7.2. 对象的引用和拷贝

对于赋值操作总是存储对象的引用,而不是这些对象的拷贝,而且对象列表,字典,元组等中存放的数据也都是数据的引用,拷贝的时候也是拷贝数据的应用,所以在对其进行处理的时候需要注意一些问题

7.3. Python 中真和假

在 Python 中任何的数据类型都可以用来表示真和假,可以通过 bool(对象)进行判断,对于表示假的值有几个固定值,除此之外,基本上都是 True 值,其中表示 False 的值有:

False 0 “” [],{},None

7.4. 对象的分类

对象类型	分类	是否可变
数字	数值	否
字符串	序列	否
列表	序列	是
字典	键值对	是
元组	序列	否
set	集合	是
range	序列	否